Departamento de Informática

PROYECTO FIN DE CARRERA

ELABORACIÓN DE UNA APLICACIÓN SOCIAL WEB 2.0 DE NOTIFICACIÓN DE MENSAJES ENTRE ALUMNOS

Autor: Ignacio Vidal Hernando

Tutor: David Palomar Delgado

Colmenarejo, Mayo de 2014

Título: Elaboración de una aplicación social web 2.0 de notificación de mensajes entre alumnos
Autor: Ignacio Vidal Hernando
Director: David Palomar Delgado

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Colmenarejo, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Durante la realización de este Proyecto Final de Carrera he recibido el apoyo y la ayuda de algunas personas entre las que me gustaría destacar:

A mi familia, especialmente a mi padre y a mi madre, por su incondicional apoyo y su infinita paciencia, y a mi hermano que ha estado apoyándome en todo momento.

A David Palomar por su ayuda y tutela en este Proyecto y por la orientación del trabajo dirigido previo que me encamino para realizarlo.

A los compañeros y amigos que he conocido durante la carrera y que han hecho que sea mucho más agradable.

Resumen

En este proyecto se ha creado un sistema de microblogging altamente interactivo mediante la tecnología Ajax. Los usuarios se validan sobre un directorio para acceder a la aplicación y una vez dentro, pueden escribir y leer mensajes cortos, enviar y recibir mensajes privados e interactuar con otros usuarios. Estas operaciones pueden realizarse de dos formas, individualmente, donde cada usuario sigue a las personas que desee y en asignaturas, que contienen a los usuarios matriculados. Estos grupos son proporcionados por el sistema automáticamente creando un grupo por cada asignatura matriculada, de esta forma se puede utilizar la aplicación como recurso docente.

Palabras clave: Microblogging; Ajax; Directorio LDAP; Base de datos; JQuery; JQuery UI; DWR; Apache DS; MySql; JBoss

Abstract

In this project, a highly interactive microblogging system has been created using AJAX technology. Users are validated through a directory to access the application and once inside, they can write and read short messages, send and receive private messages and interact with other users. These operations can be performed in two ways, one-to-one, where each user follows the people they want and in groups formed by users studiying a particular subject. These groups are created automatically by the system. A group is created for each subject, this way the application can be used as a teaching resource by the profesor.

Keywords: Microblogging; Ajax; LDAP Directory; Database; JQuery; JQuery UI; DWR; Apache DS; MySql; JBoss

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

En este primer capítulo se va dar un visión general sobre el proyecto, haciendo un resumen de lo que se verá en el resto de la memoria. Se expondrán las motivaciones y los objetivos de la realización del proyecto, así como un resumen de las fases de las que consta, lo recursos que se han utilizado para el desarrollo y un esquema de la estructura de la memoria.

1.2 Visión general

El proyecto consiste en la realización de una aplicación web de microblogging [1] que permita a los alumnos de la universidad interactuar entre ellos y con los profesores mediante el envío de mensajes de texto, consiguiendo así una herramienta docente interactiva que se una los medios de que se dispone actualmente como aula global, el correo electrónico, los foros, etc. para la interacción entre usuarios de la universidad.

Lo que se pretendía en un principio era que la aplicación se integrara con el directorio de la universidad para que todos los usuarios que estuvieran en él pudieran usar la aplicación, pero finalmente por motivos de seguridad no ha sido posible el uso del directorio real, por lo que se ha creado un directorio simulado local que, al menos, permita hacerse una idea de cómo sería el funcionamiento global de la aplicación.

La aplicación tiene varias funcionalidades, la principal, es la posibilidad de escribir mensajes de hasta 140 caracteres que aparecerán en un tablón de mensajes. Estos mensajes podrán ser leídos por otros usuarios que nos estén siguiendo. A su vez hay otras posibilidades como reescribir y responder a mensajes de otros usuarios que aparecerán también en el tablón de mensajes. Además, la aplicación tiene un apartado de mensajes privados para poder enviar mensajes privados a un usuario concreto y que no lo vean los

demás usuarios. Todos los mensajes se irán almacenando en una base de datos para mantener la consistencia del sistema y se eliminarán cuando el usuario los borre.

Otra de las funcionalidades más importantes es la posibilidad de usar la aplicación con grupos de asignaturas. Esto quiere decir que el sistema personalizará la aplicación para cada usuario dependiendo de las asignaturas en la que este matriculado, creando un grupo por cada asignatura matriculada, de este modo la aplicación se convierte en una forma de comunicación interactiva entre alumnos y profesores de las distintas asignaturas, sirviendo como herramienta para la resolución rápida de dudas que pudieran surgir en clase o cualquier otro tema a tratar, incluso la posibilidad de hacer debates, ya que los mensajes escritos en los diferentes grupos sólo podrán ser vistos por los usuarios pertenecientes a ese grupo.

Resumiendo, la aplicación tiene dos posibilidades de uso, una parte más individual, donde cualquier usuario puede seguir a otro y la parte más docente, donde se agrupan los usuarios por asignaturas.

1.3 Motivación

Lo que se quiere conseguir es una aplicación altamente interactiva que permita la comunicación entre los usuarios de la universidad.

Antiguamente era difícil hacer una aplicación web interactiva porque no se disponían de la herramientas necesarias. Las páginas web eran en su mayoría estáticas, exponían una información pero era complicado encontrar alguna que proporcionará interactividad al usuario y si había alguna era bastante lento y engorroso. Con la aparición de la web 2.0 y las herramientas de desarrollo correspondientes, las aplicaciones web son cada vez más parecidas a las aplicaciones de escritorio.

Lo que se pretende con este proyecto es estudiar, aprender y utilizar alguna de estas tecnologías para crear una aplicación altamente interactiva.

1.4 Objetivos

El objetivo fundamental del proyecto es el de crear un sistema de microblogging que esté preparado para integrarse con el directorio de la universidad y de este modo permita la interactividad entre alumnos, así como interactuar también con los profesores de las diversas asignaturas que tenga cada usuario, convirtiéndose de ese modo en una práctica herramienta como recurso docente.

Lo que pretendo con este proyecto es hacer un sistema moderno, que sea intuitivo y que permita una gran interactividad a los usuarios del sistema. Para ello me basaré en la tecnología AJAX, cuya principal característica es la interactividad, ya que cambia la forma tradicional de desarrollar aplicaciones web en donde se hacían peticiones al servidor y había que esperar a que este devolviera la nueva página para mostrar los datos. En este caso, las peticiones son asíncronas, por lo tanto se eliminan los tiempos de espera y las recargas completas de página, que podían llegar a ser muy frustrantes. Explicaré de forma más detallada dicha tecnología en capítulo 2 de esta memoria.

La aplicación que he realizado tendrá múltiples funcionalidades, pero podría generalizarse en tres principales. La primera es la escritura y visualización de mensajes de carácter general de los usuarios a los que se esté interesado en seguir. La segunda sería el envió y recibo de mensajes privados entre usuarios del sistema. Y la tercera es el envió y visualización de mensajes por grupos o asignaturas que serán obtenidas del directorio como si fuera el de la universidad.

En base al objetivo principal, se proponen los siguientes objetivos:

- Estudio de las tecnologías que componen el desarrollo mediante técnicas Ajax.
- Estudio de las librerías utilizadas para desarrollar la aplicación.
- Identificación de las funcionalidades generales y secundarias que tendrá la aplicación.
- Creación de la aplicación mediante técnicas de desarrollo web Ajax.
- Preparar la aplicación para que en un futuro pudiera integrarse con el directorio de la universidad.
- Preparación del sistema para el correcto funcionamiento en Firefox e IE.
- Conseguir un alto grado de usabilidad, tanto en el uso de la aplicación, ofreciendo ayudas a los usuarios y retroalimentación para que los usuarios no se pierdan, como en el aspecto visual, buena elección de colores, interfaz clara, etc.

1.5 Fases del desarrollo

El proyecto se divide en varias fases que comprenden una duración de 8 meses.

- Fase de estudio: la fase de estudio dura 26 días laborables. Tras la primera reunión con el director del proyecto donde se definen los objetivos generales, comienza la fase de estudio de las tecnologías con las que se desarrollará la aplicación. En esta fase también está incluido la instalación y configuración del entorno de desarrollo y las tecnologías utilizadas.
- Fase de análisis: esta fase tiene una duración de 15 días laborables. En ella se recogen los requisitos y los casos de uso del sistema.

- Fase de diseño: esta fase tiene una duración de 27 días laborables. En esta fase se obtienen los diferentes diagramas de diseño de la aplicación. Entre ellos, la arquitectura del sistema, tanto física como lógica, el diseño estático y dinámico y el diseño de la BBDD y del directorio.
- Fase de desarrollo: esta fase es la más larga del proyecto. Tiene una duración de 74 días laborables. En ella se crean las tablas de BBDD y el directorio y se implementa todo el código de la aplicación.
- Fase de pruebas: esta fase tiene una duración de 7 días. En esta fase se diseñan y realizan las pruebas finales para comprobar que los casos de uso funcionan como estaba previsto y para validar que los requisitos planteados en la fase de análisis se han cumplido.

1.6 Medios empleados

Para la realización del proyecto se ha requerido la utilización de una serie de recursos tanto Software como Hardware, que expongo a continuación.

- Software:
  - Entorno de desarrollo.
    - Eclipse: se utiliza este IDE de código abierto multiplataforma para desarrollar la aplicación. [2]
  - JDK de java : para desarrollar el código del servidor. [3]
  - Librerías.
    - DWR: es una librería Java de código abierto que permite a los desarrolladores escribir páginas web que incluyen tecnología Ajax. [4]
    - JQuery: como librería Javascript para el lado cliente. [5]
    - JQuery UI: para crear la interfaz de usuarios, ya que proporciona *widgets* y otros elementos para ello. [6]
    - JDBC: para la conexión a BBDD. [7]
    - JNDI: para conexión a directorios. [8]
  - Servidores.
    - Servidor JBoss: se utiliza JBoss como servidor de aplicaciones. [9]
    - Servidor APACHE DS: se utiliza APACHE DS como servidor de directorios. Sobre él se realiza la autenticación de usuarios. [10]
    - Servidor MySql como sistema gestor de bases de datos. [11]
  - Editores de texto.
    - Microsoft Office 2007: para realizar la documentación.
    - StarUML: para crear los diagramas UML de la fase de diseño. [12]
    - Día: para crear diagramas e ilustraciones que se incluyen en la documentación. [13]

- Navegadores web.
  - Navegador Internet Explorer: se probará la aplicación en este navegador web.
  - Navegador Mozilla Firefox: se probará la aplicación en este navegador web.

- Hardware:

  - Todo el trabajo lo he realizado en mi ordenador personal.
  - Impresora laser: para la impresión de la documentación.
  - Lápiz de memoria: para copias temporales de seguridad.
  - Disco duro externo: para el mantenimiento de copias de seguridad del proyecto.

Esto es un resumen de los elementos utilizados en el desarrollo del proyecto. Se pueden ver de forma más detallada en el capítulo 3 de esta memoria.

1.7 Estructura de la memoria

En este apartado se expone cómo está estructurada esta memoria y que contiene cada capítulo.

Capítulo 1 Introducción y objetivos.

El primer capítulo hace un resumen general de todo lo que se recogerá en los capítulos siguientes de la memoria, dando una visión general del proyecto realizado. Entre otras cosas se recogen los objetivos, las fases y los recursos utilizados.

Capítulo 2 Estado del arte.

En este capítulo se habla sobre las principales tecnologías estudiadas en las que se centra el proyecto, empezando por Ajax y continuando por las librerías que se han utilizado para el desarrollo.

Capítulo 3 Entorno de desarrollo.

En este capítulo se exponen de forma detallada los recursos hardware y software utilizados para la realización del proyecto. Incluye un apartado que explica paso a paso como instalar y configurar el entorno de desarrollo.

Capítulo 4 Análisis del sistema.

Una vez presentadas las tecnologías que se van a usar y preparado en entorno de desarrollo, en este capítulo se recogen los requisitos del sistema y los caos de uso. En este capítulo no se recogen los documentos de requisitos y casos de uso completos, simplemente es un resumen de ellos, los documentos completos se encuentran en la parte final de la memoria, en los apéndices.

Capítulo 5 Diseño del sistema.

Este capítulo recoge y explica todos los diagramas de diseño de la aplicación, empezando por la arquitectura, el diseño estático y dinámico y finalizando con el diseño de la BBDD y del directorio implementado.

Capitulo 6 Pruebas.

Este capítulo contiene las pruebas diseñadas para la comprobación del funcionamiento del sistema. También hay varias tablas que muestran la consecución de los requisitos planteados en la fase de análisis.

Capitulo 7 Presupuesto.

En este apartado se expone un resumen con la planificación del proyecto, dividiéndolo en fases y sub-fases. También se incluye el diagrama de Gantt que muestra las diferentes actividades en contexto con el tiempo. Para terminar se expone un desglose de los costes parciales y la suma total del coste del proyecto.

Capítulo 8 Conclusiones y trabajo futuro.

En este capítulo se hace una retrospectiva de la realización del proyecto, revisando los objetivos planteados al inicio y dando algunas ideas de hacia donde podrían dirigirse algunos trabajos futuros sobre el proyecto.

Glosario

Recoge los acrónimos que aparecen en la memoria con su significado.

Referencias

Este apartado recoge las fuentes bibliográficas que se referencian en la memoria. Las referencias están ordenadas numéricamente por el orden de aparición en el documento. Entre ellas hay páginas web, artículos y libros que he utilizado para documentarme.

Apéndices

Es este apartado se recogen dos apéndices con los documentos completos de requisitos de software y de casos de uso.

En el próximo capítulo se explican las bases de las tecnología Ajax que ayudarán a entender cómo funciona a más bajo nivel y se expondrán las librerías que he elegido para la realización del proyecto y como fue el proceso de su elección.

Capítulo 2

Estado del Arte

2.1 Introducción

En este segundo capítulo se van a explicar las principales tecnologías en las que se centra el proyecto.

Como se ha visto en el capítulo anterior, el proyecto consiste en la realización de un sistema de microblogging basado en la tecnología Ajax, por lo que será la primera tecnología que se explicará en este capítulo. Se comenzará describiendo que es Ajax, de que tecnologías se compone, como funciona, cuáles son sus componentes, así como sus ventajas y desventajas.

Desarrollar utilizando Ajax sin la ayuda de una librería, puede llegar a ser engorroso y complicado, por lo que para la realización del proyecto se hará uso de alguna de las librerías Ajax disponibles en la actualidad que facilitarán el desarrollo de la aplicación.

Quiero comentar que previo a este proyecto de fin de carrera, realicé un trabajo dirigido sobre la tecnología Ajax y las diferentes librerías disponibles para los desarrolladores. Después de realizar dicho estudio obtuve una serie de conclusiones en las que me he basado para elegir las librerías que me han parecido más apropiadas para la realización de este proyecto. Antes de explicar las librerías elegidas haré un breve resumen sobre dicho trabajo (librerías analizadas y conclusiones obtenidas), de este modo la consiguiente explicación estará apoyada en él.

Más concretamente explicare DWR, una librería para Java que ofrece la posibilidad de hacer llamadas remotas al servidor, JQuery y JQuery UI. Estas dos últimas son librerías Javascript. JQuery ofrece funciones para modificar el árbol DOM, captura de eventos y algunos efectos. JQuery UI proporciona funciones, widgets, estilos, efectos, etc. para la creación de interfaces de usuarios.

2.2 Tecnologías

2.2.1 Ajax

Introducción

El acrónimo Ajax viene de *Asynchronous JavaScript And XML*. Es una tecnología que permite al navegador web obtener datos del servidor y refrescar partes de una página web a través de peticiones asíncronas, es decir el cliente no tendrá que esperar a recibir la respuesta del servidor para seguir operando. Básicamente el objetivo de utilizar Ajax es realizar aplicaciones web más interactivas y dinámicas.

Más que una tecnología, Ajax es una técnica de desarrollo web para crear aplicaciones web interactivas. Además tampoco se puede considerar Ajax como una tecnología en sí misma, ya que es el conjunto de otras como HTML, CSS, Javascript, DOM, XML, trabajando conjuntamente.

Descripción

Ajax no es más que una petición HTTP [14] estándar, es decir, requiere una dirección URL a la que llamar, un método HTTP (GET o POST) y un conjunto de variables que se quieran mandar al servidor. La respuesta del servidor también es una respuesta HTTP estándar, con sus cabeceras HTTP y el cuerpo, que contendrá los datos que devuelve el servidor en formato de texto o XML. Esta llamada se hace a través del objeto Javascript XMLHTTPRequest, implementado por todos los navegadores modernos y que encapsula la comunicación HTTP de forma fácil.

La importancia de ésta técnica reside en que al utilizarla, se ejecuta en el lado del cliente, es decir, en el navegador Web, pero por debajo mantiene una comunicación asíncrona con el servidor en segundo plano, de esta manera, el usuario puede estar realizando operaciones en el navegador sin preocuparse de la comunicación con el servidor, y además es posible realizar cambios sobre la interfaz sin necesidad de recargarla entera. Esto implica un aumento en la interactividad, velocidad y usabilidad de las aplicaciones web que usan Ajax.

Modelo clásico vs Modelo Ajax

A continuación se expone y explica un diagrama que ilustrará mejor los modelos de aplicación web clásico, comparado con el modelo basado en Ajax.

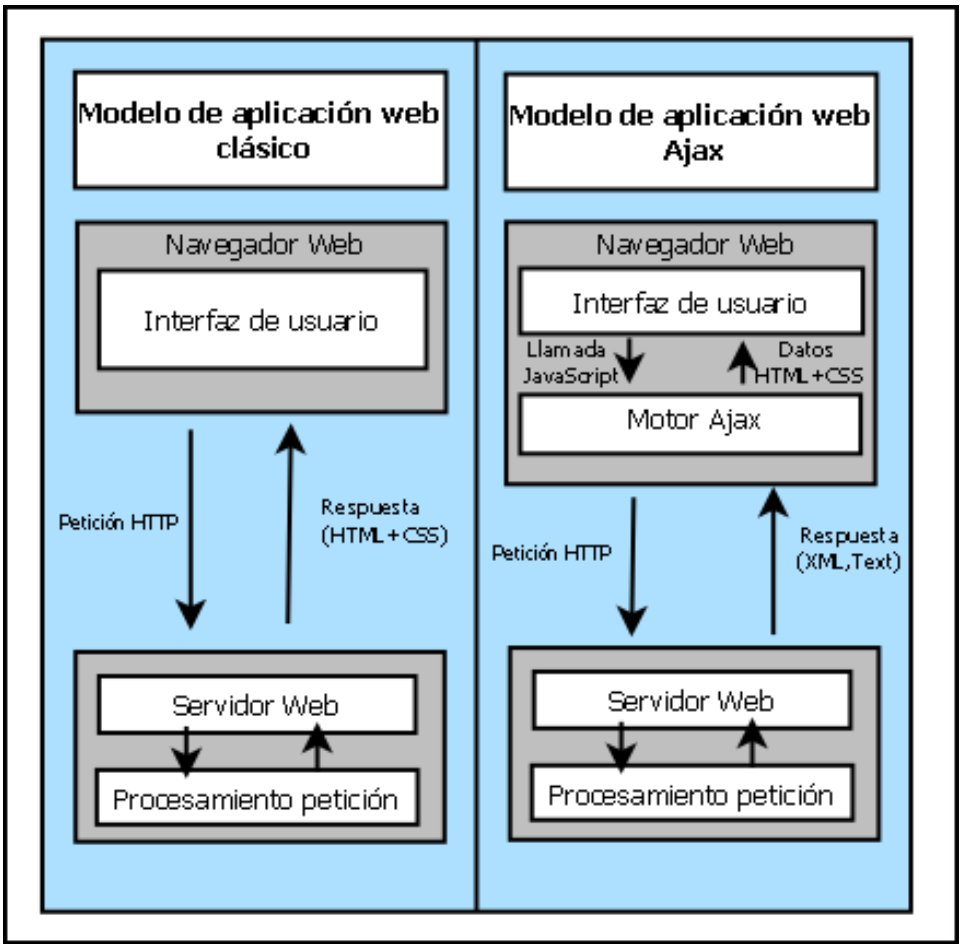


Ilustración 1. Modelos de aplicación web clásico y utilizando AJAX. (Imagen basada en la original creada por Adaptive Path **[15]**)

En el diagrama se pueden observar los dos modelos, a la izquierda el modelo clásico y a la derecha el modelo basado en Ajax.

En el modelo clásico, cuando el usuario hace la petición al servidor se queda esperando a que este procese la respuesta y se la devuelva para refrescar el navegador. En la parte inferior izquierda se puede observar que el servidor al recibir la petición procesa y opera con los datos recibidos y cuando termina envía los datos en formato HTML+CSS de vuelta al navegador, es en ese momento cuando el navegador actualiza la interfaz con los datos recibidos del servidor, recargando de nuevo toda la página.

En el modelo Ajax, el usuario hace una petición mediante Javascript pero en vez de ir directamente al servidor, va al motor Ajax que será el encargado de hacer la petición HTTP al servidor de forma asíncrona y en segundo plano. El usuario queda libre para

hacer lo que quiera en el navegador y cuando el servidor termina de procesar la petición devuelve la respuesta en formato texto o XML de nuevo al motor Ajax que se encargará de actualizar la parte de la interfaz que sea necesaria.

En resumen, con el modelo clásico el usuario tiene que esperar a que el servidor procese la petición y se recargue la página para volver a tener el control, mientras que con el modelo Ajax esto se soluciona, porque en una misma página podemos hacer varias cosas, sin tener que saltar a otra "página web", simplemente cuando el servidor devuelva la respuesta se modificará la parte de la página que sea necesario.

Mediante Ajax la interacción que tiene el usuario con la aplicación se mejora porque no tendrá que saltar de página en página para hacer alguna tarea específica, y tendrá en todo momento el control de la aplicación.

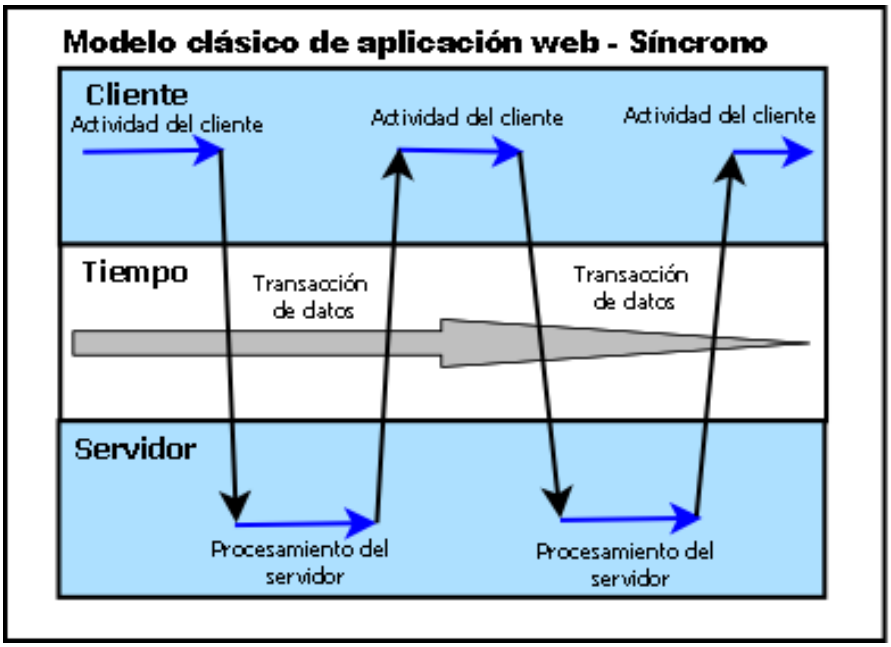


Ilustración 2. El patrón de interacción síncrona de una aplicación web tradicional. (Imagen basada en la original creada por Adaptive Path **[15]**)

En la Ilustración 2, se puede observar la interacción del usuario en el modelo clásico de aplicaciones web en donde el cliente realiza la petición al servidor y debe esperar a que este termine para recibir la respuesta y poder interactuar de nuevo con la aplicación. En la ilustración 3, se puede observar el modelo Ajax que permite que la interacción del usuario con la aplicación suceda asincrónicamente. El usuario mandará la petición al motor Ajax, que será el que se encargue de administrar la petición y de devolver la respuesta, de esta forma, el usuario nunca estará mirando una ventana en blanco del navegador o un icono de carga esperando a que el servidor procese la petición.

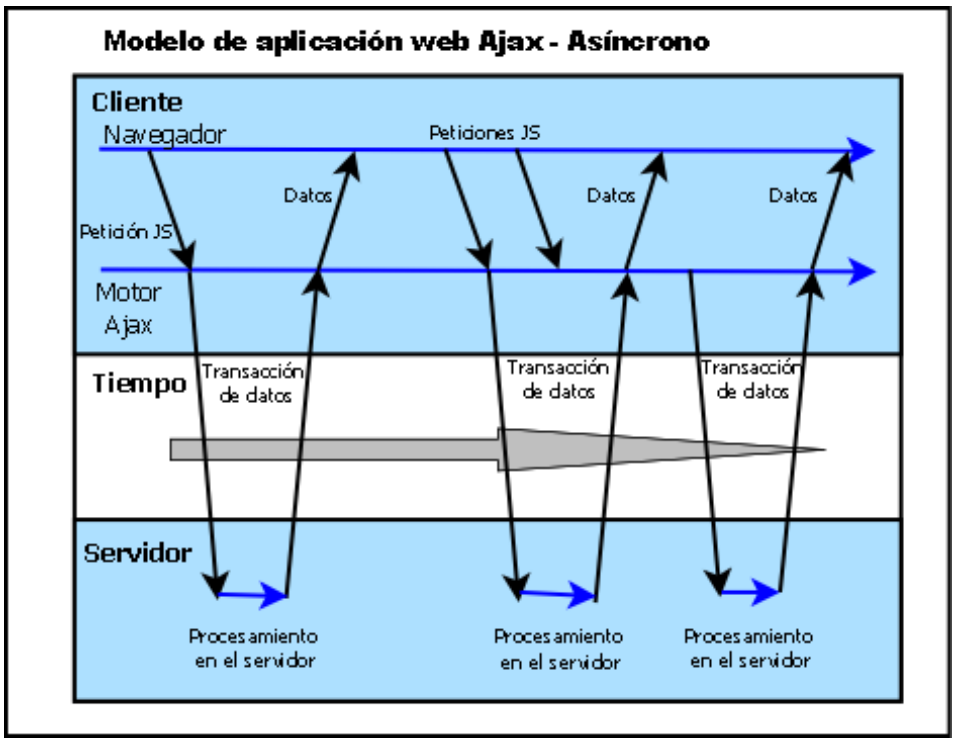


Ilustración 3. El patrón asíncrono de una aplicación Ajax. (Imagen basada en la original creada por Adaptive Path **[15]**)

Tecnologías que componen Ajax

Ajax no constituye una tecnología en sí mismo, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente:

- HTML: es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web. [16]
- CSS: es un lenguaje formal usado para definir la presentación y los estilos de un documento estructurado escrito en HTML o XML. [17]
- Javascript: es un lenguaje interpretado, es decir no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Es un lenguaje que se ejecuta en el cliente, permitirá entre otras cosas modificar, seleccionar y eliminar elementos de la página y enviar peticiones Ajax al servidor. [18]
- Modelo de objetos del documento (DOM): es una API que proporciona una serie de objetos estándar para representar documentos HTML y XML y una interfaz para manipularlos. Será accedido por el usuario con un lenguaje de scripting como Javascript, para mostrar e interactuar dinámicamente con la información presentada en las páginas web. [19]
- XML: es un lenguaje de marcas que permite estructurar documentos de forma sencilla. Es el formato más usado para la transferencia de datos con el

servidor, pero no es el único, también funcionan HTML preformateado, texto plano, etc. [20]

- El objeto XMLHTTPRequest: es el elemento principal de una aplicación Ajax que permite realizar el intercambio de datos entre la parte cliente y la parte servidor de forma asíncrona. [21]

Partes de una aplicación Ajax

Todas las aplicaciones Ajax se componen de 4 puntos, que hacen posible la comunicación asíncrona entre el cliente y el servidor. Las cuatro principales partes o piezas que forman una aplicación Ajax son:

- El Objeto XMLHTTPRequest.
- Conversación en segundo plano de la aplicación Ajax con el servidor.
- Procesamiento de los datos dentro del servidor.
- La respuesta que la aplicación Ajax recibe del servidor.

Objeto XMLHTTPRequest

Un objeto XMLHTTPRequest es una instancia de una API que nos permite la transferencia de datos desde los script del navegador (Javascript) a los del servidor (PHP, ASP, JSP, Java, etc.) e inversamente de forma asíncrona.

El objeto XMLHTTPRequest es la parte más importante de todas las aplicaciones Ajax. Su objetivo es permitir a Javascript crear peticiones HTTP y enviarlas al servidor (cuando hacemos clic sobre un enlace o enviamos un formulario, es decir, cuando ocurre algún tipo de evento) para así poder tener una comunicación asincrónica con este, de tal forma que no tengamos que actualizar la página, pudiendo seguir realizando otras operaciones.

Tiene una serie de métodos y atributos:

- **readyState**: el atributo readyState devuelve el estado actual del objeto XMLHTTPRequest. Cada vez que cambia el valor de readyState se lanza la función indicada en onreadystatechange. Los posibles estados en los que se puede encontrar el objeto son:
  1. Abierto (acaba de llamar **open**).
  2. Enviado.
  3. Recibiendo.
  4. Terminado.

  La propiedad readyState se utiliza en todas las comunicaciones asíncronas para comprobar que se puede acceder a atributos como responseXML y responseText sólo accesibles en los estados 3 y 4.

- **onreadystatechange: e**l atributo onreadystatechange asigna la función que se ejecutará cada vez que readyState cambie de valor. Frecuentemente se utiliza onreadystatechange para definir una función que tratará los datos recibidos del servidor. En este caso se comprobaría que readyState tuviera valor 4 y entonces se leería el valor de responseXML ó responseText.
- **responseText**: el atributo responseText devuelve el texto del documento descargado del servidor en una petición con XMLHTTPRequest. La propiedad responseText se utiliza para tratar los datos recibidos desde el servidor que no tienen formato XML, podremos acceder a los datos siempre y cuando el estado de la conexión devuelto con readyState sea igual a 3 (recibiendo) o 4 (terminado).
- **responseXML**: el atributo responseXML devuelve una referencia al cuerpo del documento descargado del servidor en una petición con XMLHTTPRequest en formato XML. La propiedad responseXML se utiliza para tratar los datos recibidos en formato XML, se podrá acceder a los datos siempre y cuando el estado de la conexión devuelto con readyState sea igual a 4 (Terminado). Se podrán utilizar las propiedades del Modelo de Objetos de Documento (DOM) para tratar los datos XML recibidos.
- **status**: el atributo status devuelve el código del estado HTTP de la transmisión devuelto por el servidor web. Se utiliza para comprobar que no ha habido problemas en la comunicación con el servidor.

Los métodos principales con los que cuenta el objeto XMLHTTPRequest para hacer peticiones asíncronas son:

- **Open**: el método open prepara una conexión HTTP a través del objeto XMLHTTPRequest (con un método y una URL especificados) e inicializa todos los atributos del objeto. Al llamar a open el atributo readyState toma el valor 1, además, reinicia las cabeceras de envío y devuelve a los atributos responseText, responseXML y status sus valores iniciales.

  Ejemplo de sintaxis: http.open("GET", URL, true);

  El método de conexión será ‘GET’ o ‘POST’.
  El segundo parámetro es la URL a la que queremos enviar la petición.
  El tercer parámetro indica si va a ser una conexión asíncrona o síncrona. Por definición deberemos usar modo asíncrono para que podemos llamarlo AJAX.
- **Send**: el método send envía la petición con los datos pasados por parámetro como cuerpo de la petición a través del objeto XMLHTTPRequest. Para realizar la conexión deberemos usar send después de open.

- **Abort**: el método abort detiene todas las conexiones asíncronas abiertas por el objeto XMLHTTPRequest y lo reinicializa poniendo a cero su estado (readyState).

Comunicación entre el motor Ajax y el servidor

La comunicación entre la aplicación Ajax y el servidor es simplemente el uso de los métodos que hemos visto en el apartado anterior, para que se produzcan peticiones asíncronas así como la respuesta que dará el servidor a dichas peticiones.

Para iniciar una comunicación con el servidor se utilizará en primer lugar el método open para preparar la conexión, indicándole los parámetros necesarios que se acaban de ver. A continuación, se usará la propiedad 'onreadystatechange' para indicarle qué función deberá ejecutar cuando readyState cambie de valor. Por último se utilizará el método send para iniciar la comunicación.

Problema de la memoria caché:

Una aplicación Ajax se basa en la comunicación constante entre el navegador con el servidor, por lo cual habrá que tener cuidado con la memoria caché del navegador ya que puede dar problemas a la hora de realizar una conexión asíncrona. El problema vendrá cuando el navegador quiera cargar una URL de la memoria caché, en este caso no hará la nueva petición http como debería. Para que nunca cargue de la caché habrá que asegurarse de que siempre que busque en esta, le parezca que es una nueva URL, para ello se le añade un número aleatorio a la URL de la petición (en el parámetro del método 'open'). Este problema se soluciona usando cualquiera las librerías Ajax que se expondrán en los siguientes apartados.

Procesamiento de los datos dentro del servidor

El servidor será el encargado de tramitar las peticiones que le lleguen desde la aplicación, podrá hacerlo mediante archivos PHP, Java, JSP, etc.

Tratamiento de los datos devueltos por el servidor

La respuesta del servidor podrá ser de varios tipos, en modo texto, en modo XML, en HTML preformateado entre otros. El objeto XMLHTTPRequest posee dos propiedades para tratar la información recibida. Son las propiedades que se han visto anteriormente, responseText y responseXML.

La propiedad responseText presenta los datos como un formato *string* pudiendo tratar con este mediante todos los métodos Javascript de tratamiento de strings. La propiedad responseXML provee XML compatible con el DOM, de tal forma que pueda ser analizado mediante funciones de Javascript.

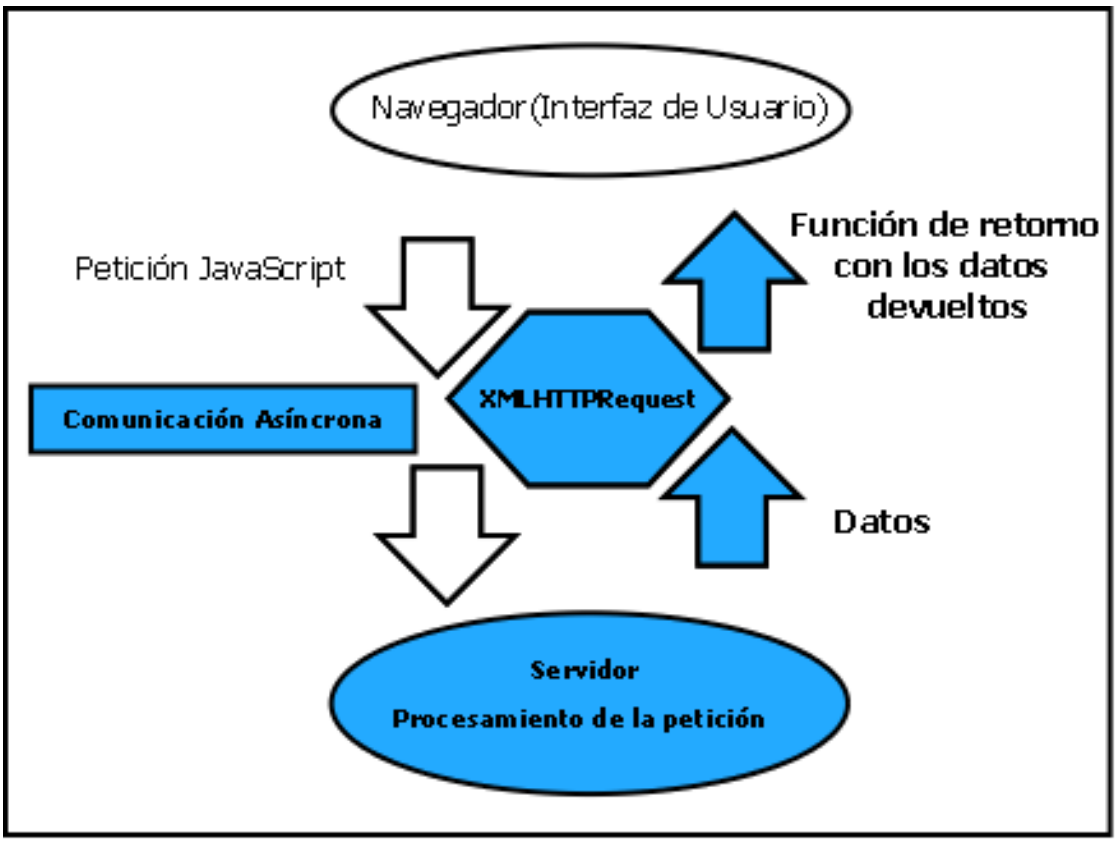


Ilustración 4. Partes de una aplicación Ajax.

Funcionamiento de una aplicación Ajax

Toda aplicación Ajax sigue una estructura básica:

- El documento HTML el cual incluye el diseño de la aplicación.
- Rutinas Javascript: estas rutinas se encargan de crear el objeto XMLHTTPRequest con el cual se podrán crear y enviar peticiones asíncronas al servidor. Además contiene la función de respuesta, la cual analiza los datos devueltos por el servidor, los trata y se los pasa al navegador para ser presentados de diversas formas. Son ejecutadas en el navegador, es decir el lado cliente.
- Rutinas del servidor: son archivos PHP, Java, etc. que procesan las peticiones que hacen los usuarios desde la aplicación Web devolviendo la información requerida. Son ejecutadas en el lado servidor.

En resumen, el funcionamiento de las aplicaciones basadas en Ajax es el siguiente:

En primer lugar el usuario provoca un evento, tras el cual se crea y configura el objeto XMLHTTPRequest. Una vez que el objeto XMLHTTPRequest se ha configurado, se envía una petición al servidor, la cual será procesada en este, mientras que el usuario puede seguir realizando las acciones que quiera en el navegador. El usuario lo único que hace es iniciar el evento. Una vez el servidor termina de procesar la petición, retorna una

respuesta que contiene el resultado. La respuesta puede ser de diferentes tipos por ejemplo un documento XML, un *array* de datos, un string de texto entre otros. Cuando se ha devuelto la petición, el objeto XMLHTTPRequest llama a la función de respuesta que procesa el resultado. Esta función será la encargada de actualizar el árbol DOM de la página asociado a la petición, con los datos obtenidos del servidor. Se actualizará únicamente la parte que corresponda de la interfaz sin tener que recargar toda la página.

Ventajas y Desventajas.

Una vez vistos los componentes y el funcionamiento de las aplicaciones basadas en Ajax se verán cuales son las principales ventajas y desventajas que aporta esta tecnología, así como que problemas resuelve.

Como ya se he comentado, Ajax se basa en un uso coordinado de distintas tecnologías que en conjunto permite una mayor rapidez y eficacia para las aplicaciones basadas en la web. El principal beneficio que esto aporta es la posibilidad de acelerar la velocidad de ejecución de las aplicaciones gracias al cambio en la forma de relacionarse entre el cliente y el servidor.

Una de las mayores ventajas que aporta la tecnología Ajax es debida al ya comentado aumento de la velocidad de ejecución de las aplicaciones web, lo que permite que estas se parezcan cada vez más a las aplicaciones de escritorio. Esto permite que los usuarios no necesiten tener instalados programas en sus máquinas, ya que pueden ejecutar la aplicación en Internet y así ahorrar recursos y memoria de sus ordenadores. Un claro ejemplo de esto es el programa "*Google Earth*" del cual hay una aplicación en la red "*Google Maps*" que si no tiene todas las posibilidades que ofrece el programa sí tiene la mayoría de ellas.

Además de este beneficio, Ajax aporta otros, pero también alguna desventaja que comentaré a continuación:

Las principales ventajas que aporta Ajax son:

- Mejora la interactividad entre el usuario y la aplicación ya que todo el intercambio y procesamiento de datos se realiza en segundo plano entre el objeto XMLHTTPRequest y el servidor.
- Se reduce el tiempo de espera del usuario ya que los datos se recuperan de forma asíncrona y no tiene que esperar a que se recargue la página completa.
- Mayor facilidad de uso para el usuario ya que no deja de ver en ningún momento la página y de esta forma no pierde el contexto, debido a que sólo se actualiza una pequeña parte de esta.
- Se reduce el tamaño de la información intercambiada, ya que sólo se necesita una parte de la página no la página completa.

- Es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como Javascript y Document Object Model (DOM).
- El usuario no tiene que instalar ningún complemento para que funcione.
- El código es público.

La tecnología Ajax también tiene algunas desventajas, como son:

- Las aplicaciones Ajax pueden resultar más difíciles de desarrollar que las aplicaciones normales.
- En ocasiones puede haber problemas para agregar a marcadores o favoritos en un momento determinado de la aplicación.
- El botón de ir a página anterior puede causar algunos problemas a los usuarios ya que los cambios realizados en la página mediante Ajax no quedan registrados en el historial y al utilizar el botón, el usuario no vuelve a un estado anterior sino que se va a la última página que visitó.
- Puede haber problemas al imprimir páginas formadas dinámicamente.
- Aunque la presentación de la interfaz se deja para el lado cliente, el servidor puede sobrecargarse si recibe varias llamadas Ajax.
- Es más complicado proveer al usuario de retroalimentación visual para indicar el estado de la petición al usuario, ya que solo actualizamos una parte de la página.
- En las versiones de Internet Explorer 6 y anteriores es necesario tener activado el ActiveX.
- Debe comprobarse la compatibilidad entre navegadores y plataformas.
- Es posible que las aplicaciones Ajax puedan no funcionar en teléfonos móviles, PDA, u otros aparatos.

Las ventajas y desventajas se han obtenido de varias fuentes. [22] [23]

Muchos de estos inconvenientes se solucionan al usar alguna de las librerías Ajax disponibles.

Compatibilidad.

Los navegadores que permiten Ajax son aquellos que están basados en el motor Gecko como Mozilla, Mozilla Firefox, SeaMonkey, Google Chrome, Nestcape versión 7.1.

También lo permite Microsoft Internet Explorer para Windows versión 5.0 y superiores, y los navegadores basados en él.

Navegadores con el API KHTML versión 3.2 y superiores, incluyendo Konqueror versión 3.2 y superiores, Apple Safari versión 1.2 y superiores.

Opera versión 8.0 y superiores, incluyendo Opera Mobile Browser versión 8.0 y superiores. [22]

Toda la información sobre Ajax que aquí expongo es un resumen de la información estudiada de diferentes fuentes. [24] [25] [26] [22] [27]

Conclusiones.

Ajax es una tecnología que cambia la forma de interactuar con las aplicaciones web, lo hace mucho más intuitivo y sencillo para el usuario. Pero hay que intentar conseguir un equilibrio entre las ventajas y desventajas que tiene. A la hora de desarrollar una aplicación en Ajax, habría que analizar lo que se quiere hacer para saber si resulta útil en cada caso. En mi opinión, las ventajas que ofrece superan en gran medida a las desventajas, pero puede haber situaciones que resulte imposible utilizarlo porque alguna de las desventajas sean insalvables para lo que se necesita desarrollar.

Toda la tecnología que he explicado en este apartado se refiere a como se desarrollaría una aplicación sin usar ninguna librería. Programar Ajax sin usar una librería es posible pero es mucho más complicado y el tiempo que se emplea es mucho mayor que si se utiliza una librería.

Para facilitarle la vida al programador existen varias librerías Javascript que encapsulan en objetos de más alto nivel todo el proceso Ajax y otras características interesantes de Javascript (elementos de arrastrar y soltar, ventanas modales, animaciones, etc.)

A continuación voy a exponer los resultados que obtuve de un trabajo dirigido sobre librerías Ajax que realicé previamente a este proyecto, cuyo objetivo fue, primero, estudiar y entender cómo funciona Ajax a bajo nivel, y segundo, estudiar y elegir las librerías que me parecieran más interesantes para la posterior realización de este.

2.2.2 Librerías Javascript Ajax. Conclusiones trabajo dirigido.

Desarrollar una aplicación Ajax con el objeto XMLHTTPRequest puede llegar a ser complicado y engorroso, sobre todo en lo que respecta a su configuración. Esto es debido a la diferencia entre los navegadores y su diferente configuración de dicho objeto para cada unos de ellos.

La principal ventaja de usar una librería es que permite a los desarrolladores centrarse en el desarrollo de las aplicaciones ocultando los detalles de implementación de Ajax, como por ejemplo el objeto XMLHTTPRequest, evitando así numerosos problemas de compatibilidad entre navegadores.

Antes de realizar este proyecto, realice un trabajo dirigido sobre las diferentes librerías Ajax que hay disponibles. El objetivo del estudio era conocer las tecnologías con las que trabaja Ajax (Javascript, XML, HTML, etc.) así como el funcionamiento de las librerías, y familiarizarse con los métodos de trabajo necesarios para el desarrollo de aplicaciones web mediante esta tecnología, para finalmente hacer una comparativa entre ellas. Para el estudio tuve en cuenta diferentes aspectos, como son la facilidad de aprendizaje y uso, rapidez de aprendizaje y desarrollo, documentación aportada, facilidad de uso de las APIs, soporte online, foros, etc.

Para que la comparativa fuera lo más objetiva posible, y estudiar todas las librerías en la misma medida, me planteé hacer un prototipo de una sencilla agenda de contactos con varias funcionalidades. De esta forma, al ser el mismo prototipo para todas las librerías, trabajaría los mismos aspectos en todas ellas.

Las librerías objetivo del estudio fueron:

- Prototype: es una librería desarrollada en Javascript por Sam Stephenson para el desarrollo sencillo y dinámico de páginas Web. Es una herramienta que implementa las técnicas Ajax. Prototype simplifica gran parte del trabajo cuando se pretende desarrollar páginas con Ajax. Es de código abierto. [28]
- Scriptaculous: usando Prototype como base, script.aculo.us se especializa en suministrar una rica experiencia al usuario con efectos animados, de arrastrar y soltar y otros componentes para realizar interfaces de usuario. Es de código abierto. [29]
- JQuery: JQuery es una librería Javascript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología Ajax a páginas web. JQuery es de código abierto, ofrece una serie de funcionalidades basadas en Javascript que simplifican en gran medida el desarrollo de aplicaciones web Ajax, tanto en tiempo como en líneas de código. [5]

- MooTools: es una librería Javascript que facilita la creación de aplicaciones web Ajax aumentando la velocidad de desarrollo y disminuyendo las líneas de código necesarias. Se compone de un conjunto de módulos que ofrecen clases de programación orientada a objetos en Javascript. Ofrece, entre otras funcionalidades, trabajo con capas, efectos, Ajax, etc. Permite desarrollar scripts en el cliente de forma sencilla y rápida sin tener que preocuparse de las particularidades de cada navegador. [30]
- Rico: Rico es una biblioteca de Javascript de código abierto para desarrollar aplicaciones ricas que utilizan Ajax. Se basa en el uso de Prototype y librerías JSON. [31]
- GWT: Google Web Toolkit es una librería creada por Google. Básicamente se basa en crear el código de la aplicación en Java con la ayuda de un entorno de desarrollo (IDE) y el compilador de la librería se encargará de traducirlo a HTML y Javascript. Permite simplificar los aspectos más complejos de la tecnología Ajax, además elimina la mayoría de los problemas que surgen de las diferencias entre navegadores ya que es compatible con varios de estos. [32]
- DWR: Direct Web Remoting, es una librería Java de código abierto que ayuda a los desarrolladores a desarrollar páginas web que incluyen tecnología Ajax. La idea básica de DWR es que permite usar funciones Java en el lado cliente como si estuvieran en el propio navegador, aunque realmente estén en el servidor de aplicaciones. Esto se hace mediante el fichero dwr.xml en el cual se le indica que clases java podremos usar en Javascript desde el navegador. Es una nueva forma de desarrollar aplicaciones que lo hace mucho más cómodo ya que hace que la comunicación asíncrona entre cliente y servidor sea mucho más natural e intuitiva que con otras librerías. [4]

Después de realizar el estudio obtuve unas conclusiones en las que me basé para la elección de las librerías con las que realizar el proyecto. En la siguiente tabla se pueden ver los resultados, valorando 3 aspectos diferentes, requisitos de aprendizaje, requisitos de desarrollo y otros requisitos.

		Prototype	Scriptaculous	JQuery	MooTools	OpenRico	GWT	DWR
Requisitos de aprendizaje	Documentación aportada en la Web oficial (Cantidad)	9	9	10	7	0	10	10
	Documentación aportada en la Web oficial (Calidad)	10	10	10	5	0	10	10
	Idioma documentación	Inglés	Inglés	IN/ES/AL/IT/PO	Inglés	-	Inglés	Inglés
	Ejemplos disponibles en la Web oficial	5	10	10	9	3	10	10
	API	Si	Si	Si	Si	No	Si	Si
	API (Calidad)	10	10	10	7	-	10	10
	Soporte online (foros, blogs, bug tracker…)	10	10	10	10	2	10	10
	Documentación en internet	10	9	10	5	1	8	9
Requisitos de desarrollo	Facilidad de uso	9	9	10	6	5	8	9
	Facilidad de aprendizaje	9	10	10	6	5	8	9
	Sencillez de programación	9	10	10	6	5	8	10
	Rapidez de desarrollo	9	9	10	8	-	9	10
	Modularidad	9	9	9	10	-	10	10
	Potencia	8	7	8	9	-	10	10
Otros requisitos	Licencias de uso	10	10	10	10	10	10	10
	Diseño de la web oficial	10	9	10	9	4	10	10
	Plug-in disponibles	10	10	10	10	6	10	-
Resultados	**Media**	**8.06**	**8.31**	**8.93**	**6.81**	**2.18**	**8.31**	**8.68**

Tabla 1. Resultados del estudio sobre librerías Ajax.

Documentación:

	Prototype	Scriptaculous	JQuery	MooTools	OpenRico	GWT	DWR
Documentación aportada en la Web oficial (Cantidad)	9	9	10	7	0	10	10
Documentación aportada en la Web oficial (Calidad)	10	10	10	5	0	10	10
Idioma documentación	Inglés	Inglés	IN/ES/AL/IT/PO	Inglés	-	Inglés	Inglés
Documentación en internet	10	9	10	5	1	8	9

Como se puede observar, las librerías con mayor cantidad de documentación son JQuery, Prototype, Scriptaculous, GWT y DWR además su calidad es muy buena, y muy fácil de entender. En este punto ganan a MooTools donde aparte de la documentación que podemos encontrar en el sitio oficial, cuesta mucho encontrar algo. Además, la calidad de la documentación no es lo alta que se podría esperar. La información que se puede encontrar sobre RICO es mínima, muy sorprendente que no tenga ningún tipo de documentación dentro de la web oficial.

Soporte online:

	Prototype	Scriptaculous	JQuery	MooTools	OpenRico	GWT	DWR
Soporte online (foros, blogs, bug tracker…)	10	10	10	10	2	10	10

La mayoría de librerías cumplen perfectamente, ofreciendo varios métodos de ayuda al programador en la web oficial.

Facilidad de uso:

	Prototype	Scriptaculous	JQuery	MooTools	OpenRico	GWT	DWR
Facilidad de uso	9	9	10	6	5	8	9

Con facilidad de uso me refiero a lo que hay que hacer para comenzar a usar las librerías. Aquí destaco las librerías en los que el peso es pequeño, y al descargar únicamente obtienes un archivo que debes introducir en la aplicación. Los mejores en este aspecto son JQuery, Prototype y Scriptaculous. También es muy fácil DWR. Un poco retrasado se queda MooTools debido a que es necesario bajarse varios módulos dependiendo de lo que quieras usar, y al principio cuesta un poco saber que modulo es el que necesitas. En cuanto a Rico, de nuevo vuelve a quedarse en última posición, ya que al descargarlo se obtiene una serie de carpetas con diferentes archivos que siendo principiante es muy difícil de entender. GWT está un poco por detrás en este aspecto ya que la configuración del IDE y la instalación del *plug-in* necesario, lo hace un poco más complicado.

Facilidad de aprendizaje y sencillez de programación:

	Prototype	Scriptaculous	JQuery	MooTools	OpenRico	GWT	DWR
Facilidad de aprendizaje	9	10	10	6	5	8	9
Sencillez de programación	9	10	10	6	5	8	10

Estos dos conceptos van muy de la mano por lo que se pueden explicar conjuntamente. En este apartado destaca por encima de todos JQuery. Las funciones que ofrece son muy sencillas de usar, es la que menos me ha costado de estudiar y de entender debido a lo intuitivo que es. También es muy fácil de programar los efectos de Scriptaculous. Prototype es bastante fácil de usar, pero lo he valorado un poco menos debido a que si no estudias primero el objeto XMLHTTPRequest puede resultar un poco difícil de entender. GWT es quizás un poco más lento de aprender y puede llegar a costar un poco el programar, sobre todo las primeras llamadas al servidor, por lo que lo valoro con un 8 en ambos casos. En cuanto a los otros dos (MooTools y Rico) me han costado bastante más, he tenido que estar bastante más tiempo para llegar a entender las funciones, ya que no son tan intuitivas. DWR también destaca es estos dos apartados, una vez que aprendes su funcionamiento es muy sencillo desarrollar aplicaciones.

Rapidez de desarrollo:

	Prototype	Scriptaculous	JQuery	MooTools	OpenRico	GWT	DWR
Rapidez de desarrollo	9	9	10	8	-	9	10

En cuanto a rapidez de desarrollo todos ellos ahorran mucho tiempo a la hora de desarrollar aplicaciones. Si tuviera que destacar alguno, volvería a ser JQuery, ya que al ser muy intuitivo y fácil de usar, se pueden implementar las aplicaciones muy rápido y sin apenas problemas. También se puede destacar la rapidez de desarrollo de GWT y DWR.

Modularidad:

	Prototype	Scriptaculous	JQuery	MooTools	OpenRico	GWT	DWR
Modularidad	9	9	9	10	-	10	10

La modularidad de todas las librerías es muy alta, por destacar alguno diré MooTools, GWT y DWR debido a que son librerías orientadas a objetos.

Tipos de licencias de uso:

	Prototype	Scriptaculous	JQuery	MooTools	OpenRico	GWT	DWR
Licencias de uso	10	10	10	10	10	10	10

La licencia de todos es de código abierto, más concretamente todas menos Rico, GWT y DWR usan la licencia MIT, además al ser código abierto su uso es gratuito. Rico usa la licencia Apache 2.0, también de código abierto, al igual que GWT y DWR.

Plug-in disponibles:

	Prototype	Scriptaculous	JQuery	MooTools	OpenRico	GWT	DWR
Plug-in disponibles	10	10	10	10	6	10	-

En internet se pueden encontrar numerosos plug-in para todas las librerías. Para trabajar con eclipse en GWT se encuentran diversos plug-in que ayudan a la creación de proyectos.

A la vista de los resultados y con la experiencia adquirida tras realizar el estudio, las librerías que he decidido usar son:

Para la parte de Ajax, es decir, la comunicación asíncrona con el servidor, he elegido DWR debido a su gran potencia, facilidad y rapidez de desarrollo.

Para el lado del cliente, tanto la interfaz de usuario como la modificación dinámica del árbol DOM he decidido usar JQuery por su facilidad de desarrollo y su magnífica documentación. Además, uniéndolo con JQuery UI se convierte en una solución perfecta para la creación de la interfaz de usuario.

Una vez introducidos las librerías elegidas para desarrollar el sistema, las explicaré de forma más detallada.

2.2.3 DWR

¿Qué es DWR?

Es una librería Java y Javascript de código abierto que permite escribir aplicaciones con Ajax. Permite al código Javascript en el navegador usar métodos Java que están en el servidor web como si estuvieran en el navegador. La principal idea de DWR es ocultar los detalles de la implementación Ajax como la configuración y las dificultades que se pueda encontrar el desarrollador con el objeto XMLHTTPRequest, de esta forma los programadores se pueden centrar en la funcionalidad de la aplicación, ahorrando mucho trabajo y tiempo. Está especialmente diseñado para trabajar con Java, es muy fácil de aprender y usar.

¿Por qué utilizar DWR?

Si no se utiliza DWR, para cada petición al servidor se tiene que crear un *servlet* [33] que recoja la petición y retorne la respuesta, por lo que finalmente una aplicación tendría gran cantidad de servlets que tendrían que ser accedidos mediante URI´s [34]. Con DWR esto no es necesario ya que permite usar los métodos de una clase Java desde el código Javascript en el navegador, tan fácil como llamar a los métodos como se haría en una clase de Java. Por ejemplo, si se quisiera acceder a varios métodos de una clase en el servidor desde el navegador sin utilizar DWR, cada uno de esos métodos necesitarían ser accedidos mediante URI's si se está usando el objeto XMLHTTPRequest o cualquier otra librería únicamente de lado cliente como Prototype o JQuery, sin embargo con DWR simplemente hay que exportar esa clase y se podrían usar todos sus métodos ahorrando mucho código y tiempo de desarrollo.

Además DWR también proporciona funciones Javascript para la modificación dinámica de la página, por lo que proporciona en una sola librería todo lo necesario para desarrollar de forma sencilla y simple una aplicación Ajax completa.

Componentes de DWR

DWR se compone de dos partes:

- Un servlet DWR Java ejecutándose en el servidor que procesa las peticiones y envía las respuestas.
- Código Javascript ejecutándose en el navegador que envía las peticiones al servidor y recibe las respuestas pudiendo actualizar dinámicamente la página con los datos devueltos.

¿Cómo funciona DWR?

DWR genera una correspondencia entre una clase Javascript y una clase Java en el servidor de forma que se podrá llamar a los métodos de dicha clase como si estuviera en el navegador aunque realmente no lo estén. La clase Javascript se encarga de manejar los detalles de la comunicación entre el lado cliente y el servidor haciéndolo transparente para el desarrollador. Entre otras cosas se encarga de la comunicación asíncrona a través del objeto XMLHTTPRequest así como de llamar a la función de retorno cuando el servidor devuelve los datos. También se encarga de convertir y pasar los parámetros enviados en las peticiones y de devolver los datos del servidor, es decir del intercambio de datos entre el lado cliente y servidor.

De forma general, el desarrollador se encargará de implementar el código del lado cliente y del lado del servidor y dejará la comunicación entre cliente y servidor a DWR.

- En el servidor:

El motor de DWR es el objeto DWRServlet. Este servlet se ejecutará en el contenedor de servlets del servidor como otro servlet cualquiera. Se encarga de la generación del código Javascript a utilizar en el cliente, las llamadas remotas a los métodos que se hayan configurado y de la conversión de tipos entre Javascript y Java, todo ello de forma transparente para el desarrollador.

Como cualquier otro servlet, hay que declararlo y mapearlo en el archivo web.xml de la aplicación web.

```
1  <servlet>
2      <servlet-name>dwr-invoker</servlet-name>
3      <servlet-class>org.directwebremoting.servlet.DwrServlet</servletclass>
4  </servlet>
5  <servlet-mapping>
6      <servlet-name>dwr-invoker</servlet-name>
7      <url-pattern>/dwr/*</url-pattern>
8  </servlet-mapping>
```

Además del archivo web.xml, también tiene importancia el archivo dwr.xml en el que se configuran las clases Java que se podrán llamar desde el lado cliente, así como los traductores de datos que se deseen. Como he comentado se pueden diferenciar dos partes en este archivo:

Declarar las clases a exportar, las cuales se definen con la etiqueta *create*. Se deben definir tres parámetros, *Class*, *Javascript* y *Creator*. Cada línea con la etiqueta create exportará la clase Java que tenga en su atributo Class para poder ser usada mediante Javascript desde el lado cliente. Se necesita el nombre totalmente cualificado de la clase Java. También hay que rellenar el atributo Javascript que será el nombre con el que se referenciará a los métodos exportados desde Javascript. En vez de llamar al método con el nombre de la clase Java se llamará a los métodos con el nombre que se le indique en

este atributo. El atributo Creator define el ámbito del objeto, normalmente se define como sesión o new.

La segunda parte son los traductores, que se definen con la etiqueta converter. Esta etiqueta tiene dos atributos, la clase a convertir y el tipo de convertidor. Para los tipos simples no es necesario definir ningún convertidor ya que ya están definidos por defecto. Existen varios convertidores incluidos en DWR, el más habitual es el converter bean, que transforma beans [35] a arrays asociativos Javascript, pero también es posible implementar convertidores propios.

```
1  <dwr>
2   <allow>
3   <create creator="new" javascript="JDate">
4   <param name="class" value="java.util.Date"/>
5   </create>
6   </allow>
7  </dwr>
```

- En el cliente:

En el lado del cliente, DWR tiene dos funciones principales. Se encarga de las llamadas a los objetos del servidor y proporciona una serie de funciones para operar dinámicamente sobre el código DHTML.

Cada vez que se exporta una clase Java del servidor para poder utilizar sus métodos desde el navegador, DWR se encarga de crear un fichero Javascript donde se encontrarán una serie de funciones correspondientes a los métodos de la clase que se ha exportado. Cada método de la clase exportada tendrá una función Javascript propia, pero todas las funciones correspondientes a los métodos de una misma clase estarán en el mismo fichero. Estos ficheros Javascript no los tiene que crear el desarrollador sino que los crea el DWRServlet directamente siguiendo la configuración del archivo dwr.xml. Además el contenido de estos ficheros es transparente, ocultan al desarrollador las posibles dificultades de la comunicación cliente-servidor del objeto XMLHTTPRequest. Cada uno de estos ficheros se puede obtener siempre de la URL <servlet-dwrpath>/ interface/<nombre clase>.js

El motor de DWR en el lado cliente se encuentra en el archivo engine.js, en el path <servlet-dwrpath>/engine.js. Es necesario importar dicho fichero para poder utilizar las facilidades de llamada a métodos remotos en el servidor, pero no es necesario invocar ninguna función de este fichero, basta con llamar a los métodos de los ficheros presentados en el párrafo anterior.

Para realizar las llamadas a los métodos remotos hay que indicar en nombre del atributo Javascript que se configuró en la exportación de la clase en el archivo dwr.xml seguido de un punto y el nombre del método de la clase al que se desee invocar. Por ejemplo si se hubiera exportado la clase Date que tuviera un método getFecha(), y se hubiera exportado con el nombre FechaJS, la llamada sería como sigue:

FechaJS.getFecha(función de retorno);

Según la filosofía Ajax que se basa en la comunicación no síncrona entre cliente y servidor, hay que tener en cuenta que a la hora de hacer las llamadas remotas, estas deben ser asíncronas, y por lo tanto hay que pasar como parámetro una función de retorno que se encargará de operar con los datos una vez devueltos por el servidor, ya sea realizando otra llamada, o actualizando la interfaz de usuario, para lo que utilizará las funciones que veremos en el siguiente párrafo. Si no se le pasa esta función la llamada no funcionará.

La segunda función que tiene DWR en el cliente es proporcionar una serie de funciones Javascript para modificar el contenido de la página que esta activa. Todas estas funciones están agrupadas en el fichero utils.js que al igual que con los demás ficheros Javascript servidos por el servlet DWR es generado automáticamente. Lo que permiten las funciones que ofrece el fichero utils.js es modificar el código DHTML, ocultando las complejidades del modelo de objetos DOM y reduciendo el tiempo y la cantidad de código que habría que escribir si se utilizará directamente código Javascript.

Entre las funciones que ofrece DWR se pueden encontrar:

- Selectores de elementos que permitirá obtener cualquier elemento del árbol DOM. Se podrá seleccionar elementos por id, clase, tipo de tag, e incluso elementos familia de otro seleccionado. Si el selector diera como resultado varios elementos los obtendría todos dentro de una array.
- Funciones para obtener y establecer el valor o conjunto de valores de un elemento. El valor o los valores que se proporcionaran a las funciones dependerán del tipo de elemento al que se aplique, por ejemplo podemos aplicar la función setValues() a un campo select, pasándole como parámetro un array con las diferentes opciones que queremos introducir. Prácticamente se pueden utilizar sobre cualquier elemento excepto tablas, listas e imágenes.
- Funciones para añadir o eliminar filas de una tabla.
- Funciones para añadir o eliminar elementos de una lista.

Estas son algunas de las funciones más utilizadas pero hay otras. La documentación completa se puede obtener en la página web de DWR [36].

A continuación y para terminar voy a exponer un resumen de cómo es el funcionamiento básico de una aplicación Ajax desarrollado con DWR.

1. Cuando la aplicación se inicia, DWR genera automáticamente las funciones Javascript correspondientes a los métodos de las clases Java que se han exportado en el fichero dwr.xml.
2. Normalmente una llamada al servidor se inicia con la activación de un evento ya sea por parte del usuario o por parte del propio sistema. Antes de llamar al servlet DWR que esta ejecutándose en el servidor, las librerías Javascript creadas recogen los parámetros y los serializan.
3. El Servlet recibe la petición y basándose en la configuración, instancia un objeto de la clase necesaria y llama al método requerido.

4. Cuando el método de la función se ha ejecutado, el valor de retorno se devuelve al servlet DWR que se encarga de serializarlo y devolverlo a la función Javascript que envió la petición.
5. La función Javascript que envió la petición recibe los datos devueltos por el servidor y llama a la función de retorno o *callback*, con estos datos como parámetro.
6. Con las funciones de la librería 'utils' se modifica la página de forma dinámica.

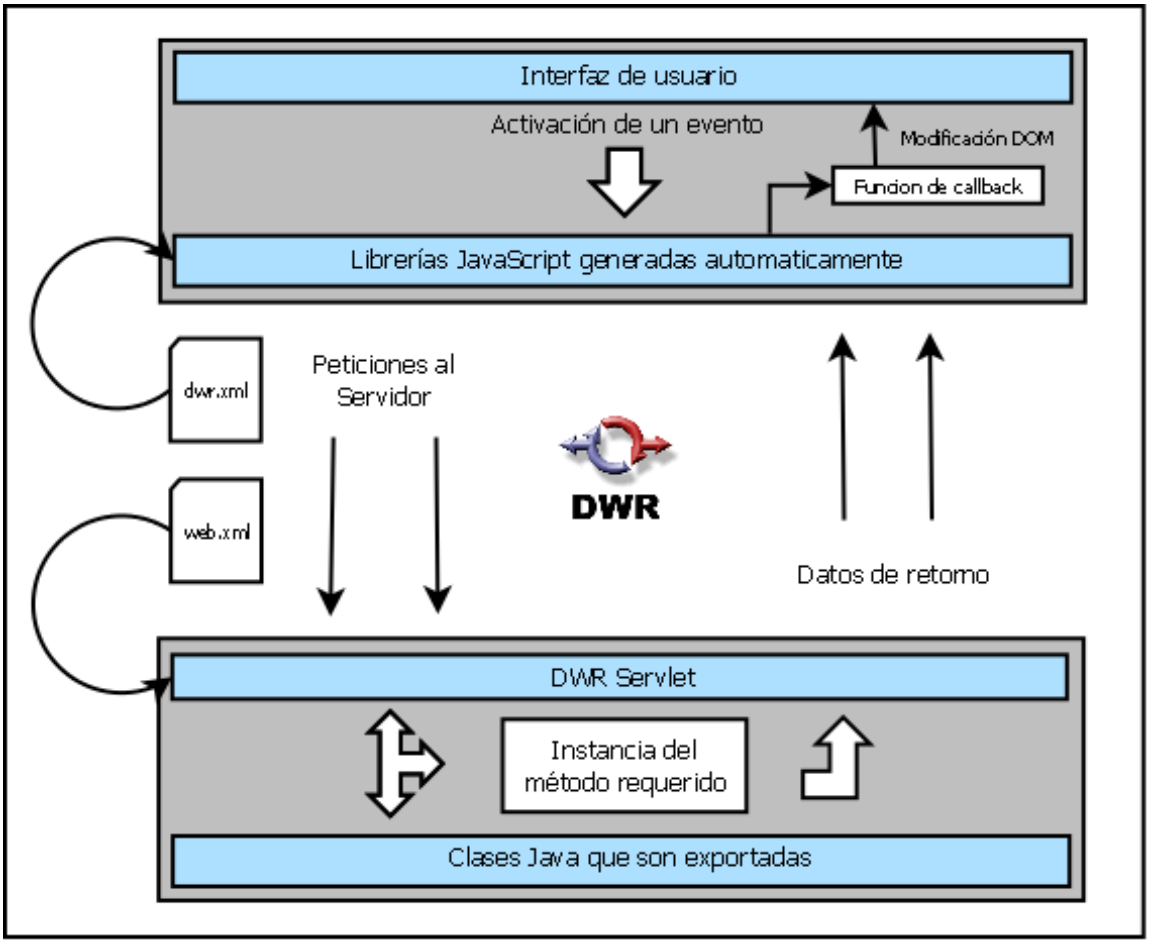


Ilustración 5. Funcionamiento de una aplicación desarrollada con DWR.

Compatibilidad

Como las aplicaciones de DWR se basan en Java, el servidor que ejecuta una aplicación DWR debe ser un servidor Java. JDK 1.3 o superior y tener la especificación de servlets 2.2 como requisitos mínimos. Casi todos los servidores de aplicaciones Java soportan las aplicaciones DWR. Apache Tomcat, Apache Geronimo, WebSphere, Weblogic, JBoss y muchos otros.

DWR soporta la mayoría de los navegadores más recientes

- Versiones de Firefox 1.x, y posteriores.
- Internet Explorer 6.0 y versiones posteriores (IE Mac no es compatible). IE 5.5 también puede funcionar, pero IE 5.0.2 no es compatible.
- Navegadores basados en Mozilla son compatibles desde la versión 1.7.
- Opera 7.5.4 y posteriores.
- Safari versión 1.2 y posteriores.

Toda la información que aquí expongo está basada en la información obtenida en el libro DWR Java Ajax Applications [37] junto con la experiencia adquirida.

Conclusiones

DWR es una librería que proporciona todo lo necesario para desarrollar una aplicación Ajax completa de forma sencilla y rápida y ahorrando gran cantidad de trabajo comparándolo con cómo sería si se desarrolla con otras librerías solo de lado cliente o por supuesto si se trabajara sin ninguna librería. En mi opinión es una librería muy recomendable por su naturalidad a la hora de implementar las llamadas asíncronas al servidor, pero para mi gusto las funciones Javascript que ofrece en el fichero utils.js para la modificación de la interfaz de usuario son menos intuitivas que las de JQuery, o simplemente que mi experiencia con esta última es mayor por lo que para esa parte utilizare la librería JQuery que explicaré a continuación.

2.2.4 JQuery

¿Qué es JQuery?

Es una librería Javascript de código abierto creada por John Resing, aunque actualmente es una librería con tanto éxito que recibe muchas contribuciones de otros programadores.

Permite simplificar la manera de interactuar con los documentos HTML, y el árbol DOM, manejar eventos, implementar animaciones y por supuesto desarrollar aplicaciones con Ajax.

¿Porqué utilizar JQuery?

Donde realmente destaca JQuery sobre otras librerías es en su sencillez y su tamaño. Es una librería Javascript con una gran sencillez, tanto de sintaxis, como de desarrollo y aprendizaje. Sus funciones le proporcionan una gran potencia, es muy fácil realizar aplicaciones de complejidad moderada de forma sencilla, ahorrando muchas líneas de código y mucho tiempo. Además es una librería muy ligera, consta únicamente de un fichero que en su versión comprimida no supera los 20KB, y para empezar a usarlo simplemente hay que importar dicho fichero.

Como he expuesto en el apartado de DWR, ofrece una librería utils.js que contiene funciones Javascript para modificar el contenido de la página que esta activa. Estas funciones permiten hacer cosas parecidas a las que ofrece JQuery, pero en mi opinión, JQuery se centra más en este tipo de funciones y es más intuitivo que DWR a la hora de programar en este aspecto. Además, la experiencia que he adquirido durante el estudio de las tecnologías es mayor usando JQuery que DWR en cuanto a la modificación del código DHTML de la página. Debido a esto para el desarrollo del sistema utilizaré DWR para la implementación de las llamadas asíncronas al servidor y JQuery para la modificación de la interfaz de usuario.

¿Cómo funciona?

A diferencia de DWR, JQuery es una librería únicamente Javascript ejecutándose en el lado cliente, por lo tanto su uso es muy sencillo, no hay más que importar el fichero mediante las etiquetas script para poder usar las funciones que provee.

Más concretamente las características de esta librería son:

- Selección, modificación e interacción de elementos DOM.
- Captura y manejo de eventos de elementos estáticos y dinámicos. Como elementos estáticos me refiero a los elementos que se crean en la carga de la página y siempre van a estar visibles, y como dinámicos me refiero a elementos que van a ser eliminados y creados dinámicamente dependiendo de las acciones que haga el usuario. Esto podría no parecer un problema, pero si no se hace bien dichos elementos podrían perder los eventos asociados.
- Manipulación de las hojas de estilo CSS.
- Proporciona efectos y animaciones aplicables sobre los elementos de la página, que ayudan a crear una interfaz más atractiva para los usuarios.
- Funciones Ajax para la realización de llamadas asíncronas al servidor. En este apartado proporciona múltiples funciones, desde funciones más complejas que permitirán configurar gran cantidad de opciones, hasta funciones más sencillas que reducen los elementos a configurar pero que para la mayoría de los casos sirven perfectamente.
- Soporta extensiones. Una extensión es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica. También se lo conoce como plug-in. Esto es muy útil ya que permite añadir nuevas funcionalidades a la librería ya sean creadas por uno mismo u obtenidas de otros usuarios.

Para ver las funciones concretas que ofrece JQuery, su página oficial consta de una API muy extensa y completa, así como ejemplos de uso y extensiones. [38]. En la web de w3schools tambien hay un apartado sobre esta librería. [39]. A su vez, existen otras fuentes de información sobre JQuery que he utilizado para estudiar la librería y realizar este apartado. [40] [41]

Compatibilidad

Es compatible con los navegadores (y sus versiones posteriores) Mozilla Firefox 2.0, Internet Explorer 6, Safari 3, Opera 10.6 y Google Chrome 8.

Conclusiones

JQuery es una tecnología que permite una mayor calidad en el desarrollo de aplicaciones web. Es una librería muy intuitiva, fácil de aprender y usar, y muy potente. Simplifica y ahorra mucho espacio y tiempo a la hora de desarrollar. A su vez, se pueden crear extensiones o usar las creadas por terceros lo que amplía mucho sus posibilidades. En mi opinión, es una tecnología que merece la pena ser tenida en cuenta para desarrollar aplicaciones web de calidad aunque por ponerle una pega, se echa en falta elementos como widgets o similares que permitan hacer más rica la interfaz de usuario. Debido a esto voy a usar conjuntamente JQuery UI que si que los proporciona y que lo explico a continuación.

2.2.5 JQuery UI

¿Qué es JQuery UI?

Es una biblioteca de componentes para la librería JQuery, que expande a esta añadiendo un conjunto de widgets, plug-in y efectos visuales para la creación de aplicaciones web altamente interactivas. Permite crear fácilmente interfaces de usuarios, con elementos y widgets que le proporcionarán un aspecto atractivo para el usuario.

¿Porqué utilizar JQuery UI?

Cuando se usa JQuery para hacer una aplicación web y es necesario realizar la interfaz de usuario, es muy recomendable utilizar conjuntamente la librería JQuery UI porque expande a JQuery con una serie de widgets y efectos visuales que permitirán crear una interfaz de usuario muy completa, con la ventaja de que ambas están desarrolladas por el mismo equipo, y por tanto se evitarán problemas de incompatibilidad. Además, al ser una expansión, la forma de desarrollar es igual, por lo tanto el utilizar la nueva librería no cuesta ningún esfuerzo adicional.

JQuery UI proporciona efectos más ricos que los de JQuery, también proporciona widgets de interface de usuario totalmente configurables y listos para usar, además de elementos de interacción como por ejemplo elementos de arrastrar y soltar que no los ofrece JQuery.

A veces resulta difícil a la hora de realizar la interfaz no cometer errores en la coherencia de los temas, por ejemplo el color de fondo con los diferentes elementos o widgets, o que todos los elementos del mismo tipo tengan la misma clase, etc. o simplemente elegir bien los colores que compondrán el tema que puedan dar lugar a errores del tipo de mala visibilidad del texto debido a la mala elección del color de fondo del elemento que lo contiene. Con JQuery UI esto se simplifica, porque en su página web se proporcionan diversos temas ya configurados con diferentes colores que se podrán descargar para usarlos directamente. La principal ventaja de JQuery UI es que cuando se descarga la librería se descarga un fichero CSS. Este fichero será diferente según el tema descargado, pero en realidad la forma de usarlo siempre es la misma ya que el nombre de las clases es siempre el mismo, por lo tanto sabiendo cuales son las características de la librería sería suficiente porque lo único que cambia es el CSS.

Para personalizar el estilo de los elementos, JQuery UI tiene disponibles unos temas por defecto que se pueden seleccionar para descargar y de esta forma facilitarnos la elección de colores. Hay una gran variedad de temas, con diferentes colores y lo suficientemente diferentes unos de otros como para no tener problemas de encontrar el tema que más se asemeje a lo que se desea. Pero si aun así no se encuentra el tema deseado, hay una herramienta online que permite crear el tema con los colores que se quieran de un modo gráfico y muy sencillo [42].

¿Cómo funciona?

Al igual que JQuery, JQuery UI es una librería Javascript que consta de un solo fichero. Para poder utilizarlo hay que exportar la librería mediante las etiquetas script, teniendo en cuenta que dependiendo de la versión que descarguemos será necesario una versión especifica de JQuery, esto es importante porque si no se utiliza la versión correcta los widgets no funcionarán. Aunque pueda parecer algo engorroso el tener que buscar la versión exacta, no lo es tanto porque cuando se descarga JQuery UI se descarga al mismo tiempo el fichero de JQuery que corresponde así como el fichero CSS del tema seleccionado. En resumen, para empezar a utilizar la librería es necesario exportar los 3 ficheros que se descargan.

Hablando exclusivamente de JQuery UI, está dividida en 4 módulos diferentes. Como es habitual en estas librerías no es obligatorio descargarse todos los módulos, de esta forma podemos descargarnos en cada momento solo el módulo que nos interese, y así optimizar el tamaño del fichero, que contendrá únicamente lo necesario.

Los cuatro módulos de los que se compone son núcleo, interacciones, widgets y efectos.

El núcleo contiene las funciones básicas necesarias para que funcionen el resto de módulos y es necesario para poder usar las funciones de la librería.

El módulo de interacciones permite añadir comportamientos básicos a los elementos.

- Draggable: añade al elemento la propiedad de poderse arrastrar.
- Droppable: asigna la propiedad a un elemento para que pueda recibir objetos soltados tras el arrastre.
- Resizable: asigna al elemento la propiedad de poder redimensionarse.
- Selectable: asigna la propiedad seleccionable a un elemento o a todos sus hijos.
- Sortable: asigna la propiedad ordenable a un elemento o a todos sus hijos.

El módulo widgets proporciona un conjunto de componentes de interfaces de usuario. Cada componente tiene un conjunto de opciones configurables y se les pueden aplicar estilos CSS. Los widgets que contiene este modulo son:

- Accordion: menú de varios paneles con efecto acordeón. Solo un panel estará activo a la vez.
- Dialog: ventanas con contenido. Hay de dos tipos, ventana no modal y ventana modal. La diferencia reside en el foco de la ventana. Las no modales permiten cambiar el foco a otras ventanas y las no modales no permiten cambiar el foco hasta que se realice alguna acción sobre la ventana.
- Slider: elemento para elegir en un rango de valores.
- Tabs: menú que tiene varias pestañas en la parte superior, las cuales podemos seleccionar individualmente para que se muestre su contenido.

- Datepicker: es una entrada para seleccionar fechas. Al pinchar sobre el elemento aparece un calendario gráfico donde podemos seleccionar la fecha que deseemos.
- Progressbar: barra de progreso. Un indicador de estado que puede ser utilizado para mostrar un porcentaje de carga por ejemplo.

El módulo de efectos es una API para añadir transiciones animadas y facilidades para interacciones. Hay efectos para ocultar y mostrar elementos, para llamar la atención del usuario, etc. Toda la información sobre las funcionalidades que proporciona, está agrupada en el apartado de documentación de su página web. [43]

Compatibilidad

Es compatible con los navegadores (y sus versiones posteriores) Internet Explorer 6.0, Mozilla Firefox 3, Safari 3.1, Ópera 9.6 y Google Chrome. [44]

Conclusiones

Es una librería simple, potente y con el concepto de widget. Además se le pueden añadir plug-in de terceros o propios, de esta forma se aumenta mucho el contenido que puede usarse.

2.3 Conclusiones

Una vez explicadas la tecnología Ajax y vistas cuales son las librerías que se utilizan en la realización de la aplicación, en el siguiente capítulo se exponen y explican los componentes hardware y software utilizados y su configuración para el desarrollo y puesta en marcha de la aplicación.

Capítulo 3

Entorno de desarrollo

3.1 Introducción

En este capítulo se van a exponer los elementos de Hardware y Software que se han utilizado para el desarrollo de la aplicación, y para la ejecución de esta. Se empezará explicando que elementos hardware se han empleado para el desarrollo, se continuará explicando el software, IDE de desarrollo, servidores, librerías y las versiones de todos ellos, y se terminará explicando cómo instalar y configurar todos los elementos para tener completamente operativo el entorno de desarrollo.

3.2 Hardware

3.2.1 Hardware de desarrollo

Para la realización de la aplicación web he utilizado mi ordenador personal cuyas características son:

Fabricante: Hewlett -Packard
Modelo: HP Pavilion dv6500 Notebook PC
Procesador: Intel® Core™ 2 Duo CPU T5450 @ 1.66GHz 1.67GHz
Memoria (RAM): 2038 MB
Tarjeta gráfica: Mobile Intel® 965 Express Chipset Family

3.2.2 Hardware de ejecución

La aplicación se ejecuta en modo local dentro de mi ordenador personal por lo tanto las características hardware son las mismas que las del apartado anterior:

Fabricante: Hewlett -Packard
Modelo: HP Pavilion dv6500 Notebook PC
Procesador: Intel® Core™ 2 Duo CPU T5450 @ 1.66GHz 1.67GHz
Memoria (RAM): 2038 MB
Tarjeta gráfica: Mobile Intel® 965 Express Chipset Family

Ratón óptico para interactuar con el sistema.

3.3 Software

En este apartado se expondrán los elementos software que se han utilizado para el desarrollo y ejecución del sistema, desde el sistema operativo del ordenador donde se ha realizado el sistema, pasando por el IDE de desarrollo, servidores, librerías y otro software de edición de texto para realizar la documentación.

3.3.1 Sistema Operativo

Windows Vista

El sistema operativo instalado en el ordenador donde he realizado el sistema es Windows Vista ™ Home Premium.

3.3.2 Entorno de desarrollo integrado (IDE)

Eclipse Helios

Como la aplicación será desarrollada en Java será necesario un IDE que permita desarrollar en Java. Eclipse no sirve únicamente para desarrollar en Java, soporta múltiples lenguajes, pero no solo eso, ya que permite trabajar con otras tecnologías asociadas instalando sus respectivos plug-in. En mi caso me he decidido a usar Eclipse por encima de otros gracias a la gran comunidad de desarrolladores de código libre que están dedicados a la implementación de mejoras del entorno, y por lo tanto se puede

encontrar gran cantidad de recursos de documentación en internet que facilita en gran medida su configuración y la resolución de los posibles problemas que puedan surgir.

La versión que he utilizado es Eclipse Helios v3.6, la versión Windows de 32 bits.

Eclipse Helios - Eclipse IDE for Java EE Developers. Se puede descargar desde la página oficial [2], bajo licencia EPL como software libre.

Plug-in instalados

Como he comentado se pueden instalar plug-in adicionales para trabajar con otras tecnologías. En mi caso para desarrollar el sistema, he instalado dos plug-in de Apache Directory, el servidor de directorios que he utilizado:

- Apache Directory Studio LDAP Browser UI, que facilita la visualización de los diferentes directorios que estén en el servidor, mostrándolos gráficamente como una jerarquía, permitiendo también ver el contenido de los elementos del directorio.
- Apache Directory Studio LDIF Editor, un editor de archivos LDIF [45], que controla los posibles errores de código de estos archivos. Los archivos LDIF son archivos de texto con la estructura de los árboles de directorios que se utilizarán para luego cargarlos en el servidor. Tienen una gramática especial por lo que este plug-in resulta muy útil.

3.3.3 Java Development Kit (JDK)

Como se va a desarrollar en Java será necesario tener instalado el JDK que provee de herramientas de desarrollo para la creación de programas en Java. En concreto he trabajado con el JDK 6 Actualización 23 .Se puede descargar desde la página web de Java [3] de forma gratuita bajo licencia GPL.

3.3.4 Servidores

Para la realización y funcionamiento de la aplicación se necesitan 3 servidores. Un servidor de aplicaciones para poder hacer las peticiones al servidor y que este ejecute la petición y devuelva las respuestas, un servidor de directorios sobre el que se validarán los usuarios para poder acceder a la aplicación y un sistema gestor de base de datos para el funcionamiento y la consistencia del sistema.

JBoss

Como servidor de aplicaciones he utilizado JBoss, sobre este servidor se ejecutará la aplicación. Se encargará de recibir las llamadas Ajax del navegador, ejecutar el código necesario y reenviar de vuelta los datos obtenidos. Es un servidor de código abierto implementado en Java por lo que podrá ser utilizado en cualquier sistema operativo que permita Java.

En concreto he utilizado JBoss v.6.0 Se puede descargar desde su página web [9] bajo licencia LPGL de forma gratuita.

MySql

Como sistema gestor de Bases de datos he utilizado MySql. Es un sistema gestor de bases de datos relacional desarrollado como software libre. Será el encargado de ejecutar las peticiones sobre la base de datos que se realizarán desde el servidor. La versión que he utilizado es MySql Server v5.1. Se puede descargar desde su página web [11] bajo licencia GPL como software libre.

También se puede descargar el driver para el servidor de aplicaciones. Un manejador que es necesario a la hora de realizar peticiones sobre MySql desde el código Java del servidor. En el apartado de instalación del entorno de desarrollo, al final del capítulo, explicaré de forma detallada los pasos que hay que seguir para su correcto funcionamiento.

El driver que he utilizado es MySql Connector/J 5.1 que se puede descargar desde la misma página.

Apache Directory

Como los usuarios se validarán sobre un directorio LDAP [46], es necesario tener instalado un servidor de directorios. Como estoy trabajando en un sistema operativo Windows he elegido un servidor que trabaja sobre dicho sistema operativo y que es software libre. Buscando un servidor con estos requisitos me encontré con un proyecto de la fundación de software de Apache que provee soluciones para directorios. Su nombre es Apache Directory y entre lo que ofrece se encuentra un servidor de directorios, Apache Directory Server, escrito completamente en Java. También ofrece una serie de herramientas para directorios desarrolladas como plug-in para Eclipse. El nombre de este subproyecto se llama Apache Directory Studio. Incluye un navegador, un editor de esquemas LDAP y un editor de archivos LDIF entre otras cosas.

Apache DS v1.5.7 Se puede descargar desde su página web [10] bajo la licencia Apache 2.0 como software gratuito.

3.3.5 Librerías

A continuación expondré las diferentes librerías que he utilizado para el desarrollo de la aplicación. Desde las librerías Java y Javascript que he utilizado para el desarrollo de la parte cliente y la comunicación asíncrona con el servidor, como las diferentes librerías para acceder a la base de datos o el directorio.

DWR

Es una librería Java y Javascript que permite realizar las llamadas asíncronas al servidor, pudiendo usar métodos de una clase Java que está en el servidor como si estuviera en el navegador. Con ella se desarrolla la parte Ajax de la aplicación.

DWR Versión 2.0.10 Se puede descargar desde su página web [4] bajo la licencia Apache 2.0 como software libre.

Lo que se descarga es un archivo jar que tendremos que localizar en el directorio WEB-INF/lib de la aplicación web a desarrollar. En el apartado de instalación del entorno de desarrollo, al final del capítulo, se explicará la forma de configurar los aspectos necesarios para que funcione correctamente.

JQuery

Es una librería Javascript que es utilizada para modificar y desarrollar la parte Javascript del navegador en el lado cliente. Permite simplificar la forma de interactuar con el código DHTML y el árbol DOM, también a manejar eventos, implementar efectos y realizar llamadas asíncronas Ajax aunque como se ha dicho en este caso las llamadas Ajax se harán con DWR.

La versión que se utiliza depende de la versión de JQuery UI ya que al descargarse esta se descarga a su vez la versión necesaria de JQuery. En concreto la versión utilizada es:

JQuery v1.5.1 Se puede descargar desde su página web [5] bajo la licencia GPL de forma gratuita.

JQuery UI

Es una librería Javascript que extiende a JQuery proporcionando nuevos efectos, elementos de interacción y widgets para la realización de la interfaz de usuario.

JQuery UI v1.8.12. Se puede descargar desde su página web [6] bajo la licencia GPL de forma gratuita.

Se puede descargar la librería con un tema predeterminado o diseñar un tema personalizado. Cuando se descarga la librería se obtienen dos archivos Javascript, uno de JQuery UI y el otro la versión de JQuery que es necesaria para trabajar conjuntamente con ambas.

JDBC

Es una librería Java creada por Sun Microsystems Inc. Es un acrónimo de Java Data Base Connectivity (conectividad a bases de datos Java), y como su propio nombre indica, proporciona a los usuarios un mecanismo de conexión a bases de datos desde código Java, además incluye clases y métodos para realizar peticiones y actualizaciones sobre una base de datos [7]. No es necesario descargarse ni instalar nada adicional ya que viene incluido en el JDK, aunque sí que hay que tener instalado el driver adecuado al sistema gestor de bases de datos que se vaya a usar ya que es el que verdaderamente implementa la funcionalidad de las clases y métodos y crea la comunicación entre el API JDBC y la base de datos real. Como ya he comentado, este driver se puede descargar fácilmente desde la página de MySql.

JNDI

Es una API de Java creada por Sun Microsystems que proporciona servicios para acceder a directorios. Ofrece un mecanismo para asociar objetos con nombres y de esta forma poder buscar objetos y datos en un directorio mediante un nombre. No es necesario instalar nada ya que al igual que JDBC está incluido en el JDK, simplemente hay que importar la librería en el fichero en el que se vayan a utilizar sus servicios. Esta API es independiente de su implementación, aunque incluye además otra API SPI (Interfaz de proveedor de servicio) que permite que las implementaciones sean integradas en el *framework* aunque deberán hacer uso de un servidor de directorios. Programar con esta librería es bastante engorroso y complicado, la documentación que hay en la página web de java es una gran ayuda. [8]

3.3.6 Editores de Texto

Microsoft Office 2007

Para la realización de la documentación se utiliza el paquete Microsoft Office 2007.

StarUML

Para la realización de los diagramas UML se utiliza StarUML. Luego se importan al documento Word correspondiente. Se puede obtener desde su página web [12] bajo la licencia GPL como software libre.

Dia

Para la realización de los esquemas generales de la parte de la documentación se utiliza el Software Dia, que permite crear diagramas sencillos de forma fácil. Se puede obtener desde su página web [13] bajo la licencia GPL como software libre.

3.4 Instalación del entorno de desarrollo.

En este apartado voy a explicar de forma general los pasos a seguir para instalar y configurar el entorno de desarrollo, el IDE, los servidores y su configuración, las librerías, etc.

Como he comentado al principio del capítulo la aplicación ha sido desarrollada en un sistema operativo Windows por lo tanto, en los siguientes comentarios se explicará cómo se instala para dicho sistema operativo.

- Como la aplicación se va a desarrollar en Java es necesario instalar el JDK de Java, que ofrece una serie de herramientas para desarrollar mediante este lenguaje. Para instalarlo simplemente hay que descargarlo de la página web y lanzar el ejecutable siguiendo los pasos que aparezcan en pantalla.
- Una vez instalado el JDK se descargará el IDE elegido, en mi caso Eclipse Helios. No hará falta instalarlo, sólo hay se descomprimir el archivo descargado en el directorio que se elija, por ejemplo directamente en la unidad C. Al iniciar por primera vez Eclipse habrá que indicar el directorio de trabajo donde se crearán todos los proyectos que se desarrollen desde Eclipse.

- A continuación habrá que instalar los servidores y configurarlos. En primer lugar se descargará el servidor de aplicaciones. El servidor JBoss no es necesario instarlo, simplemente hay que descomprimir los ficheros en un directorio. Una vez descomprimido, habrá que agregar el nuevo servidor dentro de Eclipse para poder lanzar las aplicaciones sobre él. Para ello en el apartado de servidores de Eclipse se seleccionará "agregar un nuevo servidor", y se indicará JBoss y la versión que se ha descargado. También habrá que indicar el directorio elegido donde se ha descomprimido. De esta forma queda instalado el servidor de aplicaciones, luego habrá que configurarlo para poder hacer peticiones al sistema gestor de base de datos, pero antes habrá que instalar MySql.
- Después de descargar MySql hay que instalarlo, para ello se lanza el ejecutable y se siguen los pasos que aparecen por pantalla. Durante la instalación se pedirán algunos datos de configuración básica, como por ejemplo, el número de usuarios que serán permitidos, el puerto por el que se conecta y si se quiere iniciar como un servicio de Windows o iniciarlo manualmente. A la vez de descargar MySql también se descarga el driver que se utilizará para configurar JBoss.
- Cuando está instalado MySql hay que configurar JBoss para poder hacer peticiones sobre MySql. En primer lugar hay que pegar el driver de MySql que se ha descargado dentro del directorio de JBoss, concretamente en *Jboss/common/lib/*. Este driver es el que implementa la conexión a la base de datos. A continuación hay que crear y configurar el DataSource. Un DataSource es un objeto JNDI que se encargará de conseguir la conexión a la base de datos a través del driver comentado. Es necesario rellenar una serie de parámetros para cada sistema gestor de bases de datos. En el directorio *Jboss/docs/examples/jca* del directorio del servidor JBoss hay diferentes ejemplos de DataSources para cada sistema gestor de base de datos. En este caso se obtiene el de MySql y se pega en el directorio *Jboss/server/default/deploy*. Seguidamente se rellenan los datos necesarios, nombre de JNDI, la clase del driver, la url de conexión, el usuario y la contraseña de acceso. Esto podría configurarse de la misma forma desde el propio código de la aplicación pero haciéndolo externo se evitan problemas de seguridad y sobre todo es mucho más portable.
- Por último queda instalar el servidor de directorios. Una vez descargado Apache DS, se lanza el ejecutable y se siguen los pasos que dirigirán la instalación. Durante la instalación se pedirá el directorio de instalación, el directorio donde se almacenarán los datos y el directorio donde está instalado el JDK de Java que utilizará el servidor.
  Quedaría instalar los plug-in de Apache Directory Studio para Eclipse. Se instalan como cualquier otro plug-in para Eclipse. En la pestaña *Ayuda> Actualizaciones de Software > Buscar e instalar nuevo Software* de Eclipse se selecciona *Buscar otras características para instalar* y se le indica la URL:

  http://directory.apache.org/studio/update/1.x

  Buscando en esa url se obtienen los diferentes plug-in disponibles, en mi caso, he instalado Apache Directory Studio LDAP Browser UI y Apache Directory Studio LDIF Editor. Esta información sobre la instalación de los plug-in de ApacheDS se puede ver de forma detallada desde su página web. [10]

Una vez instalado y configurado el servidor de directorios y el IDE de desarrollo, voy a explicar como se introduciría un nuevo esquema dentro del directorio. Una vez que se tiene el esquema en fichero Ldif, (para crearlo se puede utilizar el plug-in que se ha instalado), hay que modificar el fichero server.xml que se encuentra en:

ApacheDirectoryServer/instances/default/conf/

Dentro de las etiquetas <partitions> habrá que añadir un nuevo elemento <jdbmpartitions> con los atributos id y suffix. El atributo id significa el identificador de ese árbol y suffix el nodo inicial de donde se colgará todo el esquema, lo que se indique aquí debe ser igual que el nodo raíz del esquema. Después de modificar este fichero, utilizando el plug-in instalado en Eclipse es muy sencillo introducir el nuevo esquema. Lo primero es abrir la perspectiva LDAP de Eclipse, a continuación haciendo clic secundario sobre el nodo inicial example.com podremos indicarle que esquema queremos importar buscando en archivo LDIF que se desee y de esta forma aparecerá en nuevo árbol con la raíz que hemos elegido, pudiendo ver su jerarquía y sus elementos en forma gráfica gracias al plug-in instalado.

- Para terminar este apartado explico cómo se utilizan las diferentes librerías que se usan para desarrollar la aplicación. Para utilizar las bibliotecas JDBC y JNDI simplemente hay que importar la librería en los ficheros en que se vayan a utilizar.
  Para utilizar JQuery y JQuery UI hay que importarlas mediante las etiquetas <script> en la cabecera de los ficheros JSP o HTML, y de esta forma todas sus características estarán disponibles en todos los ficheros Javascript que estén importados.
  Utilizar DWR no es tan directo como las librerías anteriores aunque no conlleva ninguna dificultad. En primer lugar habrá que descargar el fichero dwr.jar e incluirlo en el directorio WEB-INF/lib de la aplicación web. A continuación habrá que definir el servlet DWR como se declararía cualquier otro servlet en el fichero web.xml de la aplicación web. A su vez habrá que crear el fichero dwr.xml para declarar las clases Java que estarán disponibles desde Javascript. En el apartado del capítulo 2 correspondiente a DWR se explica detalladamente cómo definir ambos archivos.
  Por último habrá que incluir las librerías en el fichero JSP o HTML de la página principal:

```
<script src='/[YOUR-WEBAPP-CONTEXT]/dwr/interface/[YOUR-SCRIPT].js'></script>
<script src='/[YOUR-WEBAPP-CONTEXT]/dwr/engine.js'></script>
```

  La primera serán las librerías Javascript que crea automáticamente DWR, (una línea por cada clase importada en el fichero dwr.xml) y la segunda es el motor de DWR.

  En cada una de las páginas oficiales correspondientes se puede encontrar información muy detallada de cómo poner en marcha cada uno de los componentes.

3.5 Conclusiones

Este capítulo se puede considerar una guía rápida de como instalar y configurar el entorno de desarrollo para la puesta en marcha de la aplicación, sin embargo, si se quiere más información, en las páginas oficiales correspondientes se puede encontrar información muy detallada de cada uno de los componentes.

En el siguiente capítulo se recoge un resumen de la información obtenida en la fase de análisis, donde pensé y obtuve los requisitos que tendría el sistema y los casos de uso.

Capítulo 4

Análisis del sistema

4.1 Introducción

Una vez vistos los recursos utilizados para la realización del proyecto e instalado y configurado el entorno de desarrollo, me adentro en la explicación del trabajo del proyecto. En la fase de análisis se recogen los requisitos y los casos de uso del sistema. Para evitar que se haga demasiado engorroso, en este capítulo se recoge un resumen de los requisitos y de los casos de uso obtenidos, en caso de que se quieran ver los documentos completos habrá que dirigirse al apartado de apéndices al final de esta memoria.

4.2 Definición del Sistema

Con el objetivo de poder obtener los requisitos que definirán el sistema, voy a hacer un resumen de lo que se pretende que haga el sistema.

El sistema permitirá:

- El acceso de los usuarios a la aplicación mediante un usuario y contraseña.
- La gestión de usuarios: los usuarios podrán buscar otros usuarios para seguir sus mensajes, dejar de seguirlos, bloquearlos, etc.
- La gestión de mensajes: los usuarios podrán introducir nuevos mensajes, borrarlos, responder y reescribir los mensajes de otros usuarios. Además podrán escribir y responder mensajes privados.
- La visualización de los mensajes de los usuarios a los que este siguiendo.
- Filtrar los mensajes que visualiza cada usuario mediante un formulario, que permitirá a los usuarios ver los mensajes que más le interesen según una serie de requisitos.

- Funcionamiento de la aplicación personalizado para cada usuario con sus asignaturas matriculadas.

Una vez que se tiene una visión general de lo que realizará la aplicación, se puede empezar con la captura de requisitos. A continuación expongo un resumen de los requisitos obtenidos.

4.3 Establecimiento de requisitos

Tras analizar y pensar que requisitos va a tener la aplicación paso a exponerlos con una pequeña descripción. Los requisitos obtenidos se han dividido en primer lugar en funcionales y no funcionales, y dentro de los requisitos no funcionales se dividen en requisitos de sistema, de seguridad, de usabilidad y de interfaz. Los requisitos funcionales definen las operaciones que podrán realizar los usuarios sobre el sistema y los no funcionales tienen que ver con otros aspectos.

4.3.1 Requisitos funcionales

RF-001	Autenticarse	El usuario podrá autenticarse en el sistema, de esta forma se podrá proporcionar contenido personalizado (grupos).
RF-002	Ver mensajes	El usuario podrá ver los mensajes de los usuarios a los que sigue.
RF-003	Ver resumen seguidos	El usuario podrá ver los 10 usuarios seguidos con más mensajes escritos en la página principal.
RF-004	Ver número seguidos	El usuario podrá ver el número de usuarios a los que sigue.
RF-005	Ver resumen seguidores	El usuario podrá ver los 10 usuarios que le siguen con más mensajes escritos en la página principal.
RF-006	Ver número seguidores	El usuario podrá ver el número de usuarios que le siguen.
RF-007	Ver todos seguidos	El usuario podrá ver todos los usuarios a los que sigue.
RF-008	Ver todos seguidores	El usuario podrá ver todos los usuarios que le siguen.
RF-009	Ver usuarios grupo	El usuario podrá ver los usuarios que pertenecen a un grupo cuando este viendo los mensajes de ese grupo.
RF-010	Buscar persona	El usuario podrá buscar otros usuarios del sistema por nombre o email.
RF-011	Añadir seguido	El usuario podrá añadir a otro usuario como seguido.

RF-012	Eliminar seguido	El usuario podrá dejar de seguir a un usuario al que ya estaba siguiendo.
RF-013	Bloquear seguidor	El usuario podrá bloquear a un seguidor para que este no pueda ver sus mensajes.
RF-014	Desbloquear seguidor	El usuario podrá desbloquear a un seguidor para que pueda ver sus mensajes.
RF-015	Escribir mensaje	El usuario podrá escribir mensajes nuevos.
RF-016	Escribir mensaje grupo	El usuario podrá escribir mensajes nuevos , eligiendo si lo quiere publicar en el tablón de mensajes general o en un tablón de un grupo.
RF-017	Eliminar mensaje	El usuario podrá eliminar mensajes que haya escrito.
RF-018	Reescribir mensaje	El usuario podrá reescribir el mensaje de otro usuario.
RF-019	Responder mensaje	El usuario podrá responder a otro mensaje de un usuario al que siga.
RF-020	Filtrar mensajes nombre	El usuario podrá ver los mensajes de un usuario concreto de los que sigue.
RF-021	Filtrar mensajes fecha	El usuario podrá ver los mensajes de una fecha concreta.
RF-022	Filtrar mensajes intervalo	El usuario podrá ver los mensajes pertenecientes a un intervalo de tiempo.
RF-023	Filtrar mensajes grupo	El usuario podrá ver los mensajes de un grupo al que pertenece.
RF-024	Aumentar página tablón	El usuario podrá incrementar el número de página del tablón para ver mensajes más antiguos.
RF-025	Disminuir página tablón	El usuario podrá disminuir el número de página del tablón para ver mensajes nuevos.
RF-026	Ir a página tablón mensajes	El usuario podrá ir a un página concreta del tablón para ver los mensajes de esa página.
RF-027	Enviar privado	El usuario podrá enviar un mensaje privado a otro usuario.
RF-028	Ver tablón privados	El usuario podrá ver la cabecera de los mensajes privados dirigidos a él, los mensajes que ya ha leído y los mensajes que ha enviado, en el tablón de privados.
RF-029	Leer privado	El usuario podrá leer el contenido de un mensaje privado en concreto.
RF-030	Borrar privado	El usuario podrá borrar un privado.
RF-031	Responder privados	El usuario podrá responder a un privado.
RF-032	Ver respuestas	El usuario podrá ver todas las respuestas a un mensaje determinado.
RF-033	Ver mensaje respuesta	El usuario podrá ver el mensaje al que responde otro recibido o a que mensaje ha respondido, desde el propio mensaje de respuesta.
RF-034	Usuario activo	El usuario podrá ver el nombre de usuario de la sesión con el que se conecto.
RF-035	Salir del sistema	El usuario podrá desconectarse del sistema.

RF-036	Actualización automática	Se actualizará el tablón de mensajes, el tablón de contactos y el tablón de privados automáticamente cada 25 segundos, pero sólo en el caso de que hubiera alguna modificación que ver.
RF-037	Contenido personalizado	Cuando el usuario se autentica en el sistema se le ofrece contenido personalizado, más concretamente las asignaturas en los que este matriculado, de esta forma se podrán ver los mensajes agrupados por asignaturas.

4.3.2 Requisitos no funcionales

Requisitos de sistema

RS-001	Directorio	El sistema deberá usar un servidor de directorios donde estarán almacenados los datos de los usuarios del sistema. Los usuarios se validarán sobre este directorio.
RS-002	Compatibilidad navegadores	La aplicación deberá ser compatible con Internet Explorer 7.0, Mozilla Firefox 7.0 y con versiones superiores como mínimo.
RS-003	Cumplimiento de estándares	La aplicación cumplirá los estándares de calidad de código CSS 3 de la organización W3C.
RS-004	BBDD	Se utilizará MySql 5.1 como sistema gestor de bases de datos.
RS-005	Tecnología de desarrollo	El sistema será desarrollado utilizando tecnologías Ajax (HTML, Javascript, CSS).
RS-006	Lenguaje servidor	El código del lado del servidor será desarrollado en Java con Servlet 3.0 y JSP 2.2. Se utilizará el jdk 1.6.0 para el desarrollo y la ejecución del programa.
RS-007	Librería interfaz	Para desarrollar la interfaz de usuario, se utilizarán la librerías JQuery v1.4.2 y JQuery UI v1.8.5
RS-008	Librería Ajax	Para realizar la parte de Ajax se utilizará la librería DWR v2.0.1.
RS-009	IDE	El desarrollo de la aplicación se realizará con la ayuda de Eclipse Helios.

Requisitos de seguridad

RSG-001	Sesión	Cada vez que un usuario se valide en la aplicación se creará una sesión nueva con las variables de usuario. De esta forma se conseguirá que no se pueda acceder a la aplicación sin identificarse. La sesión se cerrará cuando el usuario salga del sistema o cuando esta caduque tras 30 minutos.
RSG-002	Contraseñas cifradas	Las contraseñas almacenadas en el directorio LDAP estarán cifradas mediante el formato SHA. [47]

Requisitos de usabilidad

RU-001	Mensajes de operación	Cada vez que el usuario realice una operación sobre el sistema, este devolverá un mensaje para avisarle del resultado de dicha operación. Por ejemplo al autenticarse, al enviar un mensaje nuevo, o al añadir un nuevo usuario como seguido.
RU-002	Estilo mensajes	Se diferenciarán los tipos de mensajes para una clara, rápida e intuitiva visión de estos. Se hará mediante diferente tipo de letra, cursiva, colores, etc.
RU-003	Ayuda escritura mensajes color	El mensaje que escribe el usuario irá cambiando de color dependiendo de cuantos caracteres lleve escritos, de esta forma el usuario tendrá una ayuda visual a la hora de añadir nuevos mensajes.
RU-004	Ayuda escritura mensajes numero caracteres.	Mientras se escribe un nuevo mensaje se mostrará el número de caracteres introducidos en cada momento.
RU-005	Estilo usuarios bloqueados.	Se mostrarán de color diferente los seguidores que hayan sido bloqueados.
RU-006	Estilo URL's	Las URL de los mensajes que las contengan se mostrarán como enlaces para que el usuario pueda acceder pinchando sobre ellos. Además se le dará un color diferente dependiendo si el link ha sido visitado o no.
RU-007	Mensajes confirmación borrado.	Cuando el usuario quiera borrar un mensaje o un privado aparecerá una ventana para confirmar el borrado.

RU-008	Estilo botón privados	El botón de ver privados cambiará dependiendo de los privados recibidos sin leer, de esta forma el usuario tendrá una forma fácil de saber si tiene privados o no. El texto del botón cambiará de color cuando haya algún nuevo privado y a su vez aparecerá un número que indicará cuantos nuevos privados hay.
RU-009	Estilo tablón respuestas	El mensaje principal del que se quieren ver las respuestas tendrá un estilo diferente que destacará para hacerlo más intuitivo. Este requisito se refiere a los mensajes del tablón de respuestas.
RU-010	Instrucciones	Manteniendo el ratón sobre los elementos de la interfaz aparecerán instrucciones sobre para que sirve cada uno, de esta forma el aprendizaje del usuario es más dinámico que con un texto explicando todo el funcionamiento. Además el usuario no pierde el contexto de lo que está haciendo.
RU-011	Maquetación líquida	La maquetación de la interfaz será líquida con respecto a la resolución y al tamaño de fuente.
RU-012	Ver número página tablón mensajes	El usuario podrá ver el número de página del tablón de mensajes que está viendo.
RU-013	Ventanas modales	Para ayudar a completar actividades que no están en la pantalla principal, como ver, y enviar privados, borrar mensajes, etc. se realizarán en ventanas modales que mantienen el foco en la actividad que se está realizando, y a su vez podrá realizar la tarea sin perder el contexto.
RU-014	Mensajes de error	Cuando se produzca un error en una operación se mostrará con un mensaje, de esta forma el usuario sabrá como corregir el error, y se le ayuda a completar la tarea.
RU-015	Links externos	Los enlaces a páginas externas se abrirán en una ventana o pestaña nueva para evitar confusiones a los usuarios al perder el contexto.
RU-016	Lenguaje familiar	En toda la aplicación se utilizará un lenguaje familiar que todos los usuarios entiendan.
RU-017	Prevención de errores	Se procurará evitar los errores de los usuarios todo lo posible. Por ejemplo las fechas del formulario de búsqueda de mensajes serán *datepickers*, de forma que no puedan equivocar el formato, o el select de cambio de página en el tablón de mensajes, donde se le darán sólo las opciones de las páginas disponibles.

RU-018	Hilo respuestas privados	Cuando el usuario lea el contenido de un privado, si existen respuestas a ese privado también se mostrarán, pudiéndose ver de esa manera todo el hilo de respuestas del privado, y siguiendo mucho mejor la conversación.

Requisitos de interfaz

RI-001	Formulario autenticación	La interfaz deberá tener un formulario de autenticación con un campo de texto para el nombre de usuario y otro para la contraseña.
RI-002	Formulario nuevo mensaje	La interfaz deberá tener un textaera para introducir el texto del mensaje y un select que permita seleccionar el tablón de grupo donde se quiere publicar el mensaje.
RI-003	Formulario buscar persona	La interfaz deberá tener un formulario de búsqueda con un campo de texto y un select para poder buscar por persona o email.
RI-004	Botón eliminar seguido	La interfaz deberá tener un botón en el tablón de contactos de seguidos para dejar de seguir a un usuario .
RI-005	Botón bloquear	La interfaz deberá tener un botón en el tablón de contactos de seguidores para bloquear a un seguidor.
RI-006	Botón eliminar mensaje	La interfaz deberá tener un botón en cada mensaje propio del usuario para poder borrar el mensaje.
RI-007	Botón reescribir mensaje	La interfaz deberá tener un botón en cada mensaje de otro usuario para poder reescribir el mensaje.
RI-008	Botón responder mensaje	La interfaz deberá tener un formulario con un textarea para responder al mensaje.
RI-009	Formulario de respuesta	La interfaz de usuario deberá tener un botón en cada mensaje de otro usuario para poder responder al mensaje.
RI-010	Filtro Select usuario	La interfaz deberá tener un select de usuarios para poder filtrar los mensajes del tablón por un usuario.
RI-011	Filtro Select grupo	La interfaz deberá tener un select de grupos para poder filtrar los mensajes del tablón por un grupo.
RI-012	Filtro Input fechas	La interfaz deberá tener dos inputs donde se le indicarán dos fechas para realizar el filtro por fecha, tanto de día como de intervalo de tiempo.

RI-013	Botones paginación	La interfaz deberá tener una barra con botones que permitan avanzar a otra página, retroceder página e ir a la primera página del tablón de mensajes.
RI-014	Select paginación	La interfaz deberá tener un select que permita ir directamente a una página concreta del tablón de mensajes.
RI-015	Tabla privados	La interfaz deberá tener un elemento donde se mostrarán las cabeceras de los mensajes privados recibidos, leídos y enviados. La forma de mostrar este contenido será mediante un tablón de pestañas ya que facilita en gran medida su visualización.
RI-016	Botón enviar privado	La interfaz deberá tener un botón disponible para enviar un privado a un usuario.
RI-017	Formulario privado	La interfaz deberá tener un formulario para introducir el asunto y el texto del mensaje privado.
RI-018	Botón leer privado	La interfaz deberá tener un botón para poder leer un mensaje privado.
RI-019	Botón responder privado	La interfaz deberá tener un botón para poder responder a un mensaje privado.
RI-020	Botón eliminar privado	La interfaz deberá tener un botón para poder eliminar un mensaje privado.
RI-021	Tablón privados	La interfaz deberá tener un elemento donde se mostrará el texto de los mensajes privados que se quieran leer.
RI-022	Botón ver privados	La interfaz deberá tener un botón para ver los privados.
RI-023	Botón ver respuestas	La interfaz tendrá un botón en cada mansaje para poder ver las respuestas de cada mensaje.
RI-024	Tablón respuestas	La interfaz tendrá un elemento donde se mostrarán las respuestas.
RI-025	Enlace mensaje respuesta	La interfaz deberá tener un enlace en cada mensaje de respuesta, que permitirá ver el mensaje origen al que responde.
RI-026	Ventana respuesta	La interfaz deberá tener una ventana donde se mostrará el mensaje al que responde otro.
RI-027	Botón salir	La interfaz deberá tener un botón para poder salir del sistema.
RI-028	Elemento usuario	La interfaz deberá tener un elemento que muestre el nombre del usuario identificado en la sesión, en la parte superior derecha de la página.
RI-029	Página única	La interfaz del sistema estará formada por dos pantallas. La pantalla de autenticación, y la página principal. En la página principal se realizarán todas las funciones.

RI-030	Interfaz Ajax	Se implementará una interfaz Ajax (con botones de acción, tablones que serán rellenados dinámicamente, etc.) que proporcionará control directo a los usuarios.
RI-031	Barra de navegación	La interfaz tendrá en la parte superior una barra donde se incluirán botones para operaciones especiales, como ver el tablón de privados o buscar usuarios, de esta forma el usuario tendrá más fácil acceder a estas actividades.
RI-032	Barra lateral	La interfaz tendrá en la parte derecha una barra lateral en la que se incluirá un textarea con el que se introducirán nuevos mensajes y el tablón de contactos.
RI-033	Botones de acción	Se usarán botones para representar las acciones que tendrán un diseño gráfico claro para que se distingan bien que son botones que se pueden pulsar. Además las etiquetas serán claras y en el caso de solo ser iconos se presentará información adicional.
RI-034	Botón seguir usuario	La interfaz tendrá un botón para seguir a un nuevo usuario.
RI-035	Botón ver seguidos	La interfaz tendrá un botón para ver los usuarios a los que se está siguiendo.
RI-036	Botón ver seguidores	La interfaz tendrá un botón para ver los usuarios seguidores.
RI-037	Reloj de sesión	La interfaz tendrá un reloj de cuenta atrás, que indicará al usuario cuanto tiempo le queda antes de que se cierre la sesión.

Para obtener los requisitos de usabilidad me he basado en el libro The Design of Sites: Patterns for Creating Winning Web [48] que proporciona una serie de directrices para crear aplicaciones web.

La definición completa de los requisitos del sistema se puede encontrar en el documento de requisitos de software, correspondiente al apéndice A al final de este documento.

A continuación se muestra un resumen de los casos de uso del sistema.

4.4 Casos de uso

Los casos de uso definen las operaciones que pueden realizar los usuarios sobre el sistema. Más concretamente, los casos de uso definen los pasos que deben seguir los usuarios para realizar las operaciones sobre el sistema.

Para comenzar se definen los actores que intervienen en el sistema, en mi caso, sólo hay un actor general que podrá realizar todas la operaciones.

Los casos de uso obtenidos son los siguientes:

- Autenticación.
- Administrar usuarios.
  - Buscar usuarios.
  - Seguir usuario.
  - No seguir usuario.
  - Bloquear usuario.
  - Desbloquear usuario.
  - Ver seguidos.
  - Ver seguidores.
- Administrar mensajes.
  - Administrar mensajes públicos.
    - Escribir mensaje.
    - Borrar mensaje.
    - Reescribir mensaje.
    - Responder mensaje.
  - Administrar mensajes privados.
    - Enviar privado.
    - Leer privado.
    - Borrar privado.
    - Responder privado.
- Visualizar mensajes.
  - Leer mensajes.
  - Filtrar mensajes.
  - Modificar página tablón.

Algunos de los casos de uso los he agrupado para que los diagramas queden más claros. A continuación voy a exponer dichos diagramas. En ellos se pueden ver los casos de uso y los actores.

El primer diagrama es un diagrama general donde se agrupan algunos de los casos de uso para que se vea menos sobrecargado. El resto son la descomposición de estas agrupaciones.

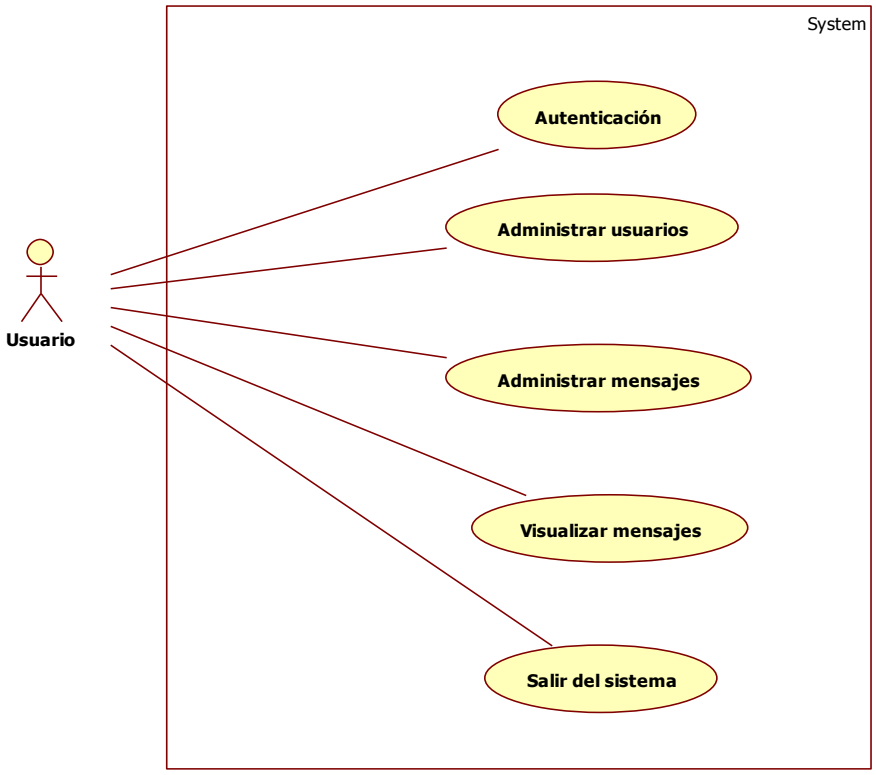


Ilustración 6. Diagrama de casos de uso general

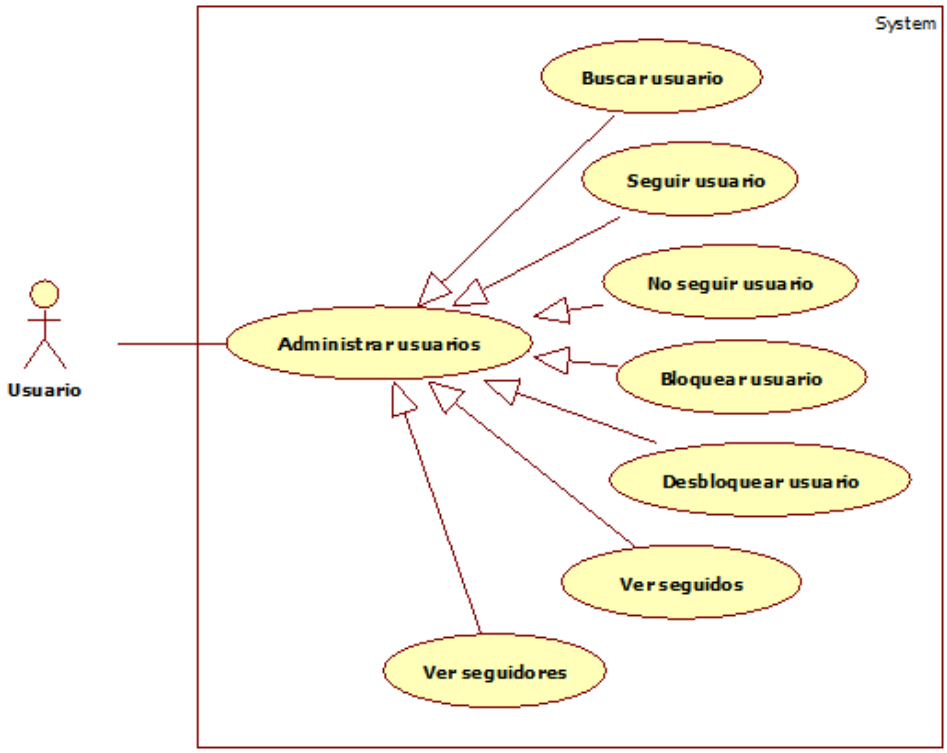


Ilustración 7. Diagrama de casos de uso "Administrar usuarios".

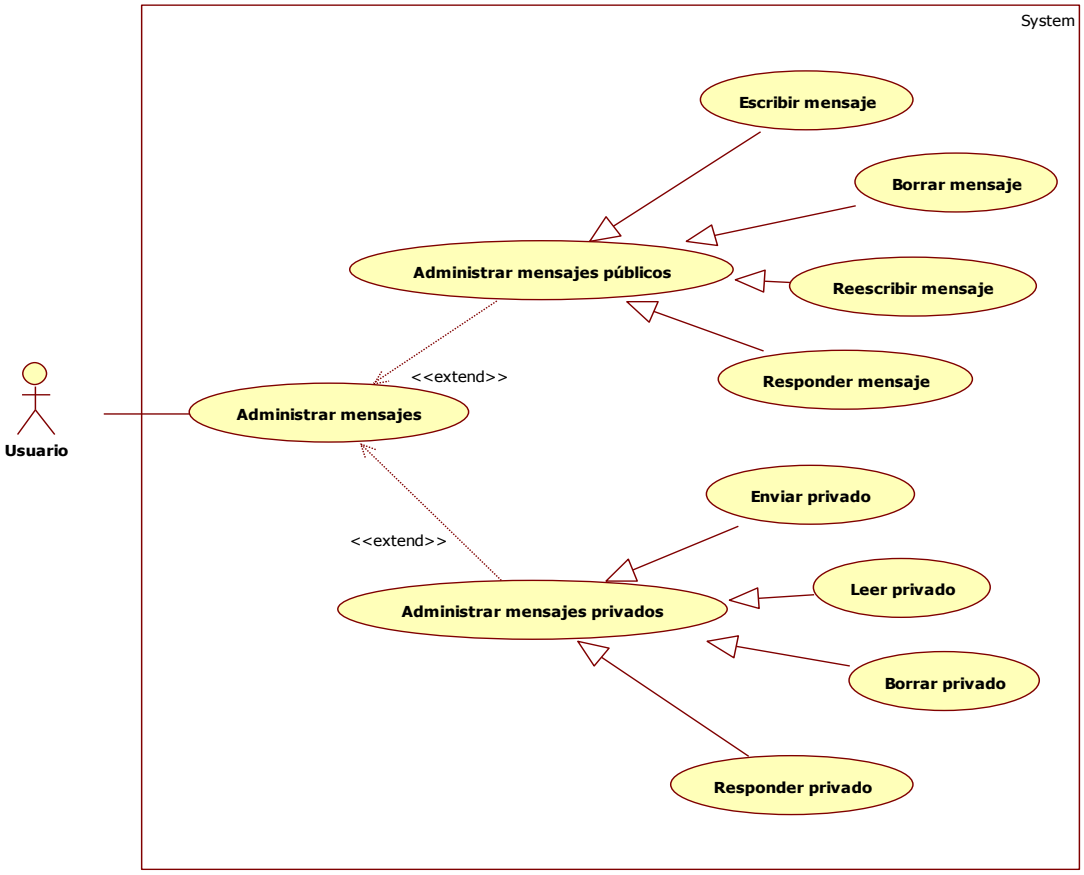


Ilustración 8. Diagrama de casos de uso "Administrar mensajes"

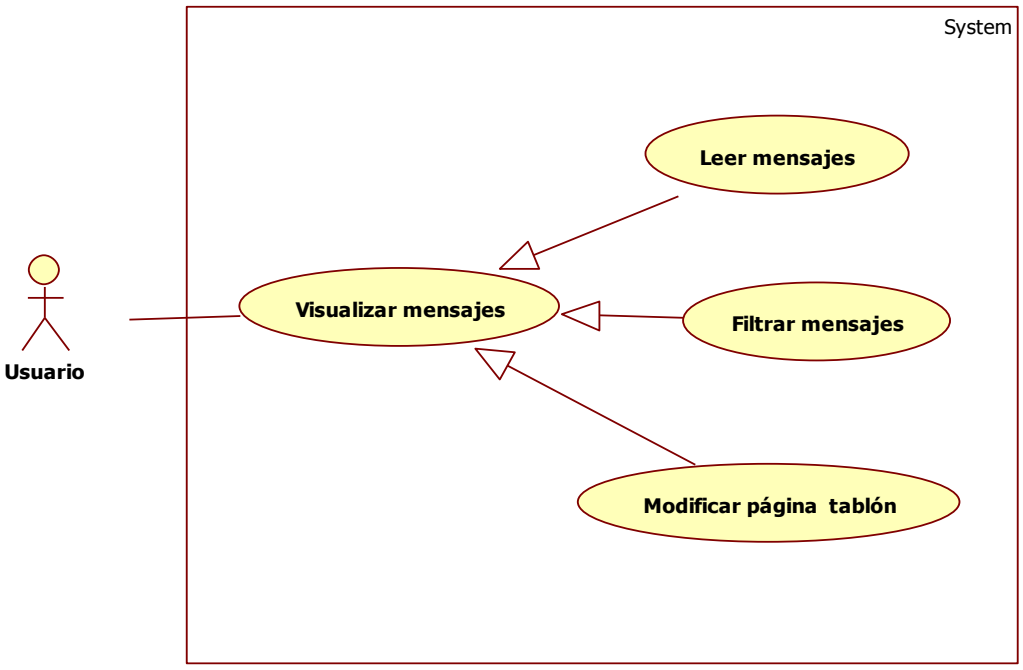


Ilustración 9. Diagrama de casos de uso "Visualizar mensajes".

4.4 Conclusiones.

Con los diagramas de casos de uso termina este capítulo. Si se quieren ver los casos de uso extendidos con los pasos que realizan los usuarios para llevarlos a cabo, habrá que dirigirse al apartado de apéndices, concretamente al apéndice B donde está el documento de casos de uso completo.

Una vez terminada la fase de análisis, donde se han obtenido los requisitos y los casos de uso del sistema, se puede empezar con la fase de diseño. En el siguiente capítulo se van a exponer la arquitectura del sistema, el diseño estático y dinámico y el diseño tanto de la base de datos como del directorio.

.

Capítulo 5

Diseño del sistema

5.1 Introducción

Una vez realizado el análisis donde se han obtenido los requisitos y los casos de uso del sistema, se va a explicar en qué consiste el diseño de la aplicación. Comienzo explicando la arquitectura física y lógica, continuo explicando el diseño estático y dinámico y termino exponiendo la estructura de la BBDD y del directorio. Para desarrollar la aplicación se va a adoptar una arquitectura de tres niveles, que es una especialización de la arquitectura Cliente-Servidor, esto se explicará de forma más detallada en los puntos siguientes.

5.2 Arquitectura del sistema

La aplicación adoptará una arquitectura de tres niveles. Esta arquitectura es una especialización de la arquitectura de dos niveles o cliente-servidor. En la arquitectura cliente servidor, se separa la carga de cómputo en dos capas, la parte de cliente y la parte de servidor aunque sin una división muy clara de las funciones de cada una. [49]

En el caso de la arquitectura de tres niveles o capas, la capa del servidor se divide en dos, separando los datos en una capa diferente, de forma que finalmente quedaran tres capas bien diferenciadas.

- Capa de presentación, es la interfaz de usuario. En esta capa se presenta la aplicación a los usuarios, permitiendo a estos comunicarse con el sistema, en sentido de entrada mediante peticiones al servidor, o de salida mostrando la información generada al usar la aplicación. Esta capa sólo se comunica con la de negocio.
- Capa de negocio, se encarga de recibir las peticiones de los clientes, gestionarlas y devolver las respuestas a los usuarios. Se comunica con la capa de presentación para

recibir y devolver las peticiones de los usuarios y con la capa de datos para obtener, actualizar o borrar datos de la base de datos.

- Capa de almacenamiento de datos. En esta capa se almacenan los datos para mantener la persistencia del sistema. Se encargará de recibir las peticiones sobre la base de datos, obtenerlos y devolverlos. Únicamente se comunica con la capa de cálculo. Esta capa contiene un sistema gestor de base de datos que se encargara de gestionar las peticiones mencionadas.

5.2.1 Arquitectura Física

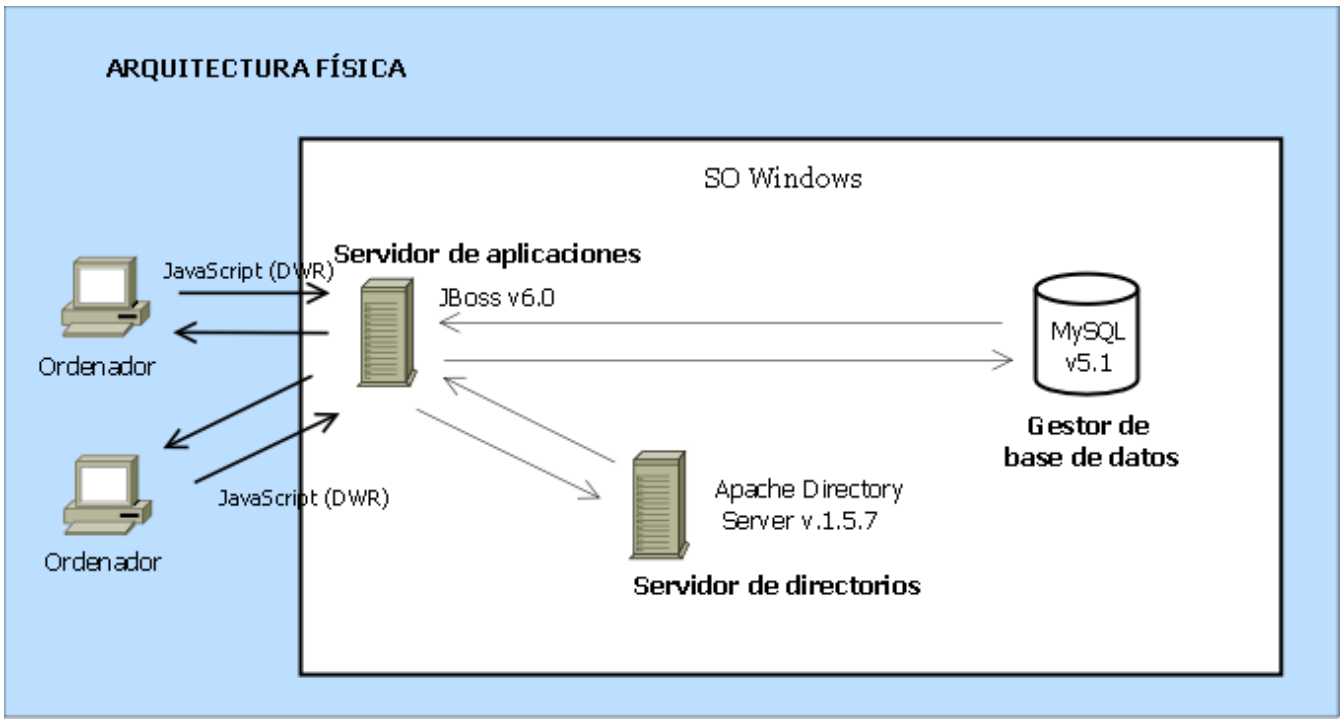


Ilustración 10. Arquitectura Física.

La aplicación estará instalada en un servidor de aplicaciones JBoss sobre el que se ejecutará. Para el almacenamiento de los datos generados por los usuarios al interactuar con la aplicación se utilizará el sistema gestor de base de datos MySql. Los datos de acceso e información personal de cada usuario estarán almacenados en un directorio. Para poder acceder, los usuarios tendrán que validarse sobre ese directorio con sus credenciales. En un principio la aplicación se integraría con el servidor de directorios de la universidad pero por motivos de seguridad no se ha podido acceder a él directamente, por lo que se ha creado un esquema simulado para el caso. Este directorio estará configurado en el servidor de directorios Apache DS.

La aplicación desarrollada se ejecuta en un entorno local que en este caso será mi ordenador personal, en él estarán instalados los tres servidores comentados en el párrafo anterior.

Funcionamiento general de la aplicación:

Los usuarios se conectarían a la aplicación, que estará ejecutándose en el servidor de aplicaciones (JBoss), mediante sus ordenadores personales a través de un navegador web.

Una vez conectados al servidor de aplicaciones se les mostrará la página de entrada de la aplicación, donde los usuarios proporcionarán sus credenciales a través de peticiones DWR para poder acceder a ella.

La aplicación se conectará al servidor de directorios (Apache DS) para comprobar si puede acceder, en cuyo caso afirmativo obtendrá los datos personales correspondientes a dicho usuario, para mostrarlos en la página principal.

Una vez que el usuario se ha conectado a la aplicación, se recupera de la base de datos (MySql) la información relacionada con el uso de la aplicación para ese usuario.

Por cada acción que realice el usuario la aplicación accederá a la base de datos ya sea para recuperar información, para guardar nueva o eliminarla.

Los elementos concretos sobre los que se apoyará la aplicación son:

- Java v.6: la aplicación ha sido desarrollada en java por lo que es necesario tener instalado el JDK 6 que proporciona las herramientas necesarias para desarrollar y ejecutar la aplicación.

- Servidor de aplicaciones (JBoss v.6.0)

  Se utilizará JBoss como servidor de aplicaciones sobre el que se ejecutará la aplicación, que se encargará de recibir las peticiones de los clientes, realizar las operaciones necesarias y devolver las respuestas.

- Sistema gestor de base de datos (MySql v.5.1)

  Se utilizará el sistema gestor de base de datos MySql para guardar los datos necesarios en la base de datos y de esta forma mantener la consistencia del sistema.

- Servidor de directorios (ApacheDS v1.5.7)

  Los clientes se validarán sobre un directorio, por ello se utilizará ApacheDS como servidor de directorios, que se encargará de obtener las credenciales y comprobarlas para dar acceso a los usuarios.

- Librerías

  DWR v2.0.10: la librería DWR permitirá realizar las peticiones asíncronas al servidor de aplicaciones y de recibir la respuesta de igual forma.

  JQuery v1.5.1: la librería JQuery permitirá modificar y gestionar los elementos DOM de la interfaz de usuario, así como gestionar y manejar los eventos de usuario.

  JQuery UI v1.8.12: la librería JQuery UI proporciona widgets y estilos para la creación de la interfaz de usuario.

  JDBC: esta librería contenida en el JDK de Java, permite obtener las conexiones a la base de datos y proporciona funciones para realizar las peticiones de lectura y escritura sobre ella.

  JNDI: esta librería proporciona una serie de servicios para conectar a directorios.

Una vez explicada la arquitectura física se expone la arquitectura lógica que da una idea de cómo se estructura los diferentes módulos de la aplicación.

5.2.2 Arquitectura Lógica

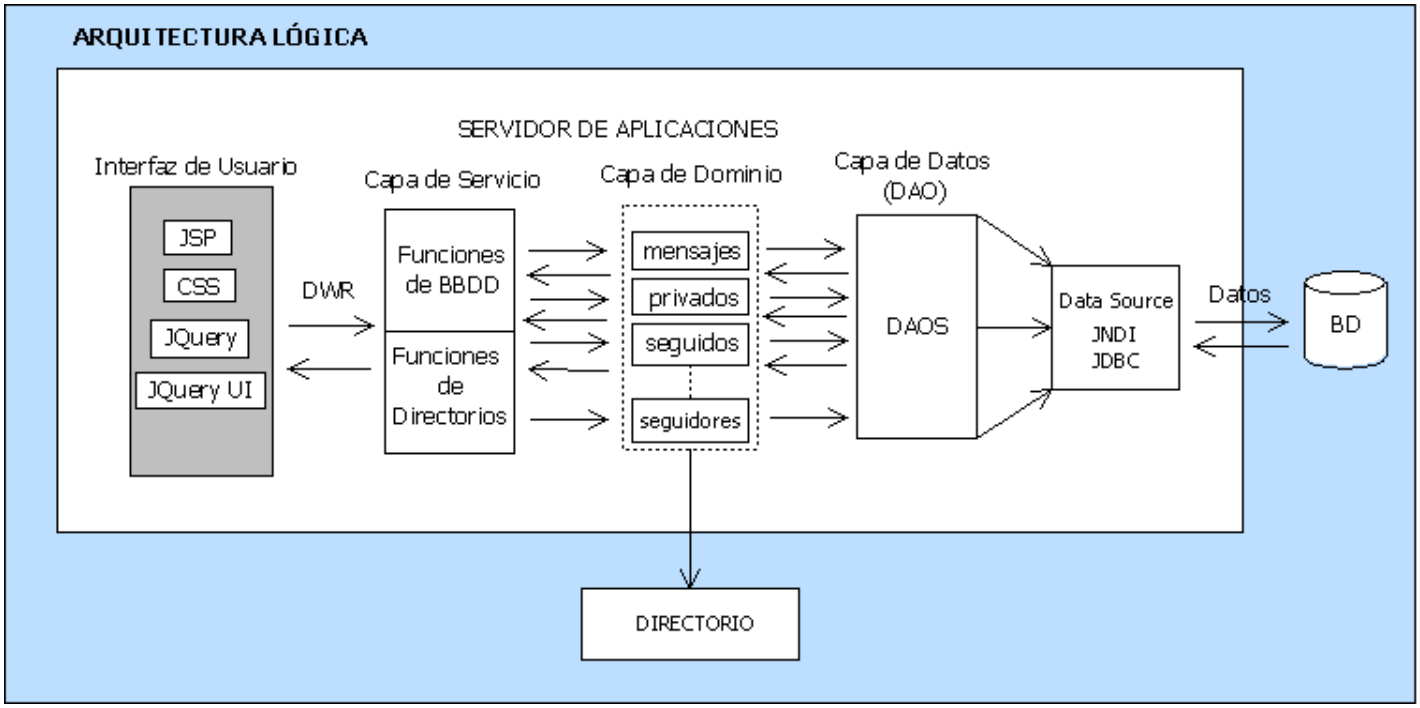


Ilustración 11. Arquitectura Lógica.

La arquitectura lógica de la aplicación estará compuesta de tres módulos o capas. La capa de servicio, la capa de dominio y la capa de datos o capa DAO.

La primera, es la capa de servicio donde estarán todas las funciones que harán funcionar al sistema. Cada acción del usuario conllevará una petición al servidor, la cual lanzará una función en dicho módulo. Las llamadas a dichas funciones se realizarán de forma asíncrona desde el lado del cliente, mediante Javascript a través de la librería DWR. Estas funciones Java podrán usarse en el navegador mediante Javascript gracias a la librería DWR, y son las que realizan todas las funciones de la aplicación, como por ejemplo la actualización del tablón de mensajes, la gestión de mensajes, la gestión de los usuarios, la gestión de los mensajes privados, etc. También contiene las funciones relacionadas con el directorio para la validación de los usuarios y la gestión de grupos.

La capa de dominio contiene diferentes clases Java que se corresponderán con cada una de las tablas de la base de datos de la aplicación. Cada clase tendrá el mismo nombre que su tabla correspondiente y tendrá los atributos que tenga cada tabla, de esta forma, la capa de servicio puede usar las clases de esta capa para el envío y la recepción de datos a la tercera capa. Cada vez que haya que pasar algún dato desde la capa de servicio a la capa DAO, se instanciará un objeto de esta capa, se rellenarán los atributos necesarios y finalmente se pasará el objeto como parámetro, de esta forma se evitan pasar parámetros sueltos.

La tercera capa es la capa de datos o capa DAO (Data Acces Object), que proporciona una interfaz de comunicación entre la capa de servicio y el sistema gestor de base de datos. [50] Habrá una clase DAO por cada tabla de la base de datos. Estas clases se encargarán de obtener la conexión a la base de datos mediante JDBC a través de un DataSource [51] y cada una tendrá sus propios métodos que realizarán las peticiones sobre su correspondiente tabla. Separar esta capa de la capa de negocio tiene como objetivo dotar de mayor flexibilidad al sistema, ya que los objetos de negocio no necesitan saber el destino final de los datos, a su vez, si se quisiera cambiar la tecnología que accede a los datos, al ser independiente de la capa de negocio sólo habría que retocar la capa DAO.

A su vez, el código de las *querys* o peticiones a la base de datos será independiente del código de la aplicación. Esto se consigue mediante un fichero properties [52]. Un fichero properties es un archivo de texto que contendrá pares de valores, con un identificador y un valor, que suele usarse para almacenar los parámetros de configuración de una aplicación. De esta forma desde el código de los métodos de las clases DAO que harán uso de ellas, solo habrá que importar la query correspondiente, indicando el nombre del identificador del fichero properties. Es aconsejable hacerlo de este modo porque de esta forma, además de generalizar el código, si se desea cambiar alguna de las querys no es necesario modificar el código, simplemente habría que abrir el fichero donde estarán todas las querys y modificar el par concreto.

En resumen, desde el punto de vista lógico, el funcionamiento de la aplicación es como sigue. La interfaz de usuarios de la aplicación es un archivo JSP corriendo en el servidor de aplicaciones. Cuando los usuarios realizan cualquier acción, se enviará una petición a la capa de servicio mediante Javascript. La capa de servicio gestionará la petición recibida mediante la ejecución de una de las funciones que contiene esta capa.

Normalmente cada acción conllevará la inserción, modificación o eliminación de algún o algunos datos en la base de datos. Para ello la capa de servicio se valdrá de las clases de la capa de dominio para el paso de datos a la capa DAO que será la encargada de obtener la conexión y realizar la consulta sobre la base de datos a través de un DataSource usando la librería JDBC.

Para que se diferencien bien las capas de la arquitectura del sistema nombradas en el apartado 5.2 de las capas de la arquitectura lógica, expongo el siguiente diagrama con el objetivo de que quede más claro.

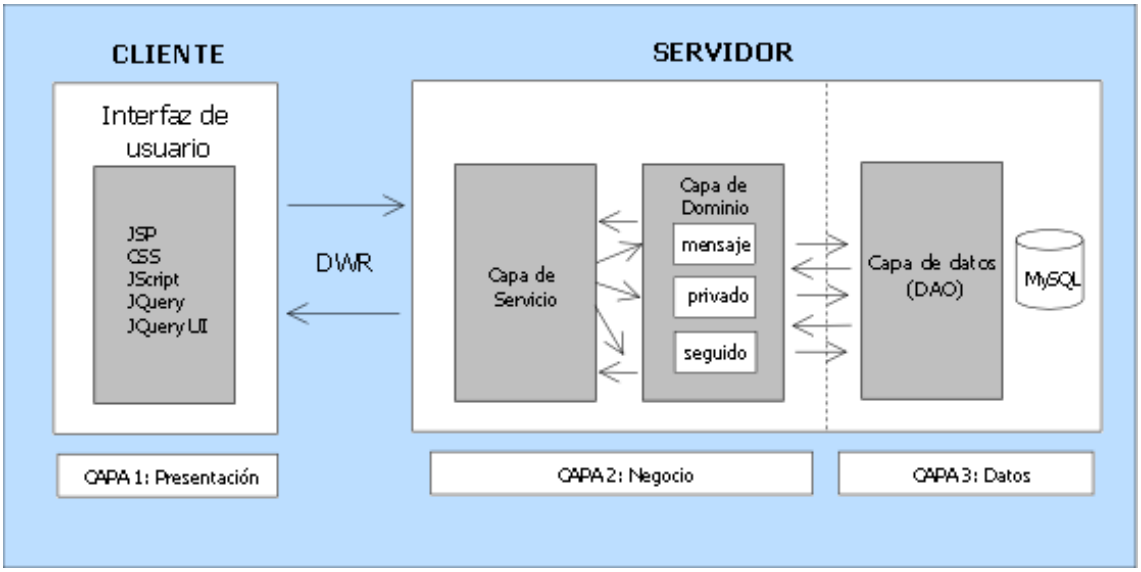


Ilustración 12. Arquitectura del sistema - Arquitectura Lógica.

5.3 Diseño estático.

5.3.1 Modelo de clases de diseño.

En esta sección se van a exponer las clases, atributos y métodos que compondrán el sistema, además de explicar cuáles son sus funciones y objetivos. Finalmente se incluirá el diagrama de clases que mostrará la conexión entre estas.

Como he comentado en el apartado anterior, el sistema está dividido en tres capas. La capa de servicio representa el punto de entrada para los usuarios y proporciona todas las funciones que estos necesiten a la hora de manejar la aplicación. La capa de dominio es una capa auxiliar que contiene las clases que sirven para la comunicación de la capa de servicio y la capa de datos, y la última es la capa de datos que contiene las clases que realizan la conexión a la base de datos y al directorio LDAP y modifican, añaden o eliminan los datos necesarios.

5.3.1.1 Capa de Servicio

Servicio.java

Proporciona los métodos que utilizan los usuarios para realizar las operaciones sobre el sistema. Estos métodos serán los que se exportan a través de DWR para que puedan ser utilizados desde el lado cliente. Los métodos de esta clase, hacen uso de los métodos de las clases de la capa de datos usando las clases de dominio para hacer la transferencia de los datos necesarios.

Atributos:

- servidor: es una cadena que contiene el nombre del servidor de directorios. Inicializado como 'localhost'.
- dn_base: la raíz del directorio de donde cuelga el árbol de datos.

Métodos:

Métodos sobre usuarios, autenticación y LDAP

- primerLogin: comprueba si el usuario ha entrado antes a la aplicación o es la primera vez.
- completarLogin: en caso de que sea la primera vez que el usuario entra en la aplicación, obtiene datos adicionales del directorio y los añade a la base de datos a la tabla de usuario para que el sistema pueda funcionar correctamente.
- comprobarActualización: comprueba si debe actualizarse el tablón de mensajes. Si la fecha de última actualización del tablón del usuario es

menor que la de última actividad de los usuarios a los que sigue, deberá actualizarlo.

- act_fec_tablon: actualiza la fecha de actualización del tablón de mensajes cada vez que este se recarga.
- login: valida los usuarios sobre el LDAP para darles acceso a la aplicación.
- obtenerAtributos: obtienen los datos necesarios del LDAP, para actualizar la tabla de la base de datos correspondiente al usuario que se conecta.
- rellenarSelectGrupos: obtiene los grupos a los que pertenece el usuario del directorio para rellenar el campo select de grupos de la interfaz de usuarios.
- rellenarSelectUsuariosGrupo: obtiene los usuarios pertenecientes a un grupo, del directorio para rellenar el campo select de usuarios de grupo de la interfaz de usuarios.
- compararFechas: compara dos fechas e indica cuál de ellas es mayor.

Métodos sobre mensajes

- nuevomsg: permite introducir un nuevo mensaje en el sistema.
- borrarMensaje: permite eliminar un mensaje del tablón de mensajes.
- aTablonNuevo: actualiza el tablón de mensajes, es decir obtiene los menajes de la base de datos que debe mostrar.
- reescribirMensaje: permite reescribir un mensaje de otro usuario. Aparecerá en el tablón como un mensaje propio pero con la referencia al usuario que lo escribió.
- responderMensaje: permite responder al mensaje de un usuario.
- obtenerIdRespondido: obtiene el contenido de un mensaje concreto. Se utiliza para mostrar el mensaje al que responde otro.
- aTablonUsuario: actualiza el tablón de usuarios con los mensajes de un usuario concreto.
- aTablonFecha: actualiza el tablón de mensajes teniendo en cuenta las fechas del filtro de búsqueda de mensajes.
- aTablonGrupo: actualiza el tablón de mensajes de un grupo determinado.
- aTablonRespuestas: actualiza el tablón de respuestas de un mensaje. Este tablón muestra todos los mensajes que responde a otro. Es un tablón diferente al principal, aparece y desaparece según quiere el usuario.
- GrupoMensaje: obtiene el grupo al que pertenece un mensaje.
- convertirUrl: transforma los links en modo texto que aparecen en un mensaje en un link que pueda pincharse.

Métodos sobre privados

- nuevoPrivado: permite escribir un nuevo mensaje privado.
- aPrivados: obtiene los mensajes privados de un usuario para actualizar el tablón de privados nuevos, leídos o enviados. En el tablón solo se muestra una línea por cada privado, para leer el contenido habría que hacer clic encima.
- leerPrivados: permite leer el contenido de un privado.

- marcarLeido: marca como leído un mensaje privado cuando el usuario lo ha abierto, modificando el campo correspondiente en el privado de la base de datos.
- obtenerJerarquiaPrivados: obtiene los identificadores de los privados que forman una conversación privada para poder mostrarlos todos cuando se abra un mensaje privado.
- borrarPrivado: permite borrar un mensaje privado.

Métodos sobre seguidos

- buscarPersona: permite buscar usuarios del sistema dados un nombre o un email.
- seguirUsuario: permite añadir como seguido a otro usuario del sistema.
- compruebaSeguidor: comprueba si el usuario al que se quiere seguir ya está siendo seguido o no.
- aSeguidos: actualiza la tabla de 5 seguidos.
- aSeguidores: actualiza la tabla de 5 seguidores.
- borrarSeguido: permite dejar de seguir a otro usuario.
- contarContactos: cuenta el número total de seguidos y seguidores que tiene un usuario.
- rellenarSelectUsuarios: rellena el campo select del filtro de búsqueda con los usuarios a los que se está siguiendo, teniendo en cuenta si algún usuario ha bloqueado al usuario conectado, en tal caso no aparecerá.

Métodos sobre bloqueados

- bloquearUsuario: permite bloquear a un usuario para que no pueda ver nuestros mensajes.
- desbloquearUsuario: permite desbloquear a un usuario para que pueda volver a ver nuestros mensajes.

- obtenerFecha: obtiene la fecha del sistema. Se utiliza para determinar la hora en la que se produjo alguna operación como por ejemplo escribir un mensaje, actualizar el tablón de mensajes, etc.

Clase Servicio.java

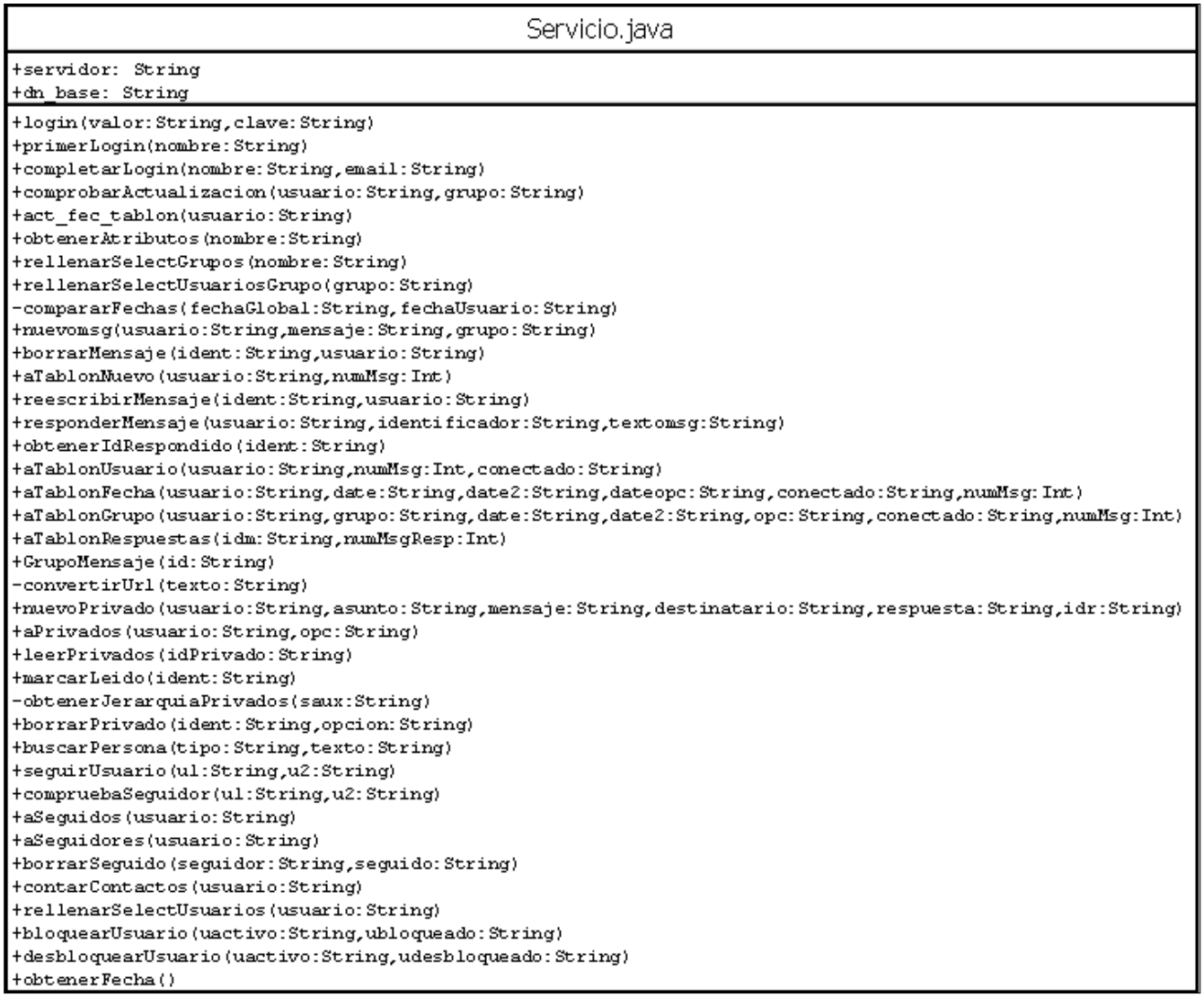


Ilustración 13. Clase Servicio.java

5.3.1.2 Capa de Dominio

Todas las clases de esta capa tienen su tabla correspondiente en la base de datos. El objetivo de esto es facilitar la transmisión de datos de la capa de servicio a la capa de datos y viceversa. Cada clase tiene tantos atributos como campos tiene su correspondiente tabla de la base de datos. Los métodos de estas clases son los 'get' y 'set' para establecer y obtener los valores de dichos atributos.

Usuario.java

Es una clase que representa a un usuario del sistema.

Atributos

- n_completo: nombre completo de un usuario, diferente del identificador de usuario.
- n_usuario: el identificador de un usuario en el sistema.
- pass: clave de entrada a la aplicación.
- email: email de un usuario.
- fecha_act: fecha de última actividad de un usuario. Cualquier actividad que pueda tener que ver con cambios para los otros usuarios del sistema.
- f_ult_tablon: la fecha de la última actualización del tablón de mensajes. Comparando esta fecha con la de actividad de otros usuarios se podrá saber si hay que actualizar de nuevo el tablón.
- num_mensajes: número de mensajes pertenecientes a un usuario. Con este valor se ordenarán los usuarios de más a menos activos.

Métodos

Los métodos de esta clase son los set y get para cada atributo.

Mensaje.java

Es una clase que representa a un mensaje de un usuario.

Atributos

- id: identificador del mensaje. Cada mensaje tiene un identificador único.
- usuario: identificador de usuario del autor del mensaje.
- texto: texto del mensaje.
- fecha: fecha de escritura del mensaje.
- reescrito: campo que indica si es un mensaje reescrito o no.
- autor: identificador del usuario autor del mensaje original.
- fecha_re: fecha del mensaje original reescrito.
- respuesta: campo que indica si es un mensaje respuesta o no.
- num_respuesta: identificador del mensaje al que responde el mensaje actual.
- destinatario: identificador del usuario al que va dirigida la respuesta.
- grupo: identificador del grupo donde se va a escribir el mensaje.

Métodos

Los métodos de esta clase son los set y get para cada atributo.

Privado.java

Es una clase que representa un mensaje privado.

Atributos

- id: identificador del mensaje privado. Cada mensaje privado tiene un identificador único.
- autor: identificador del usuario autor del mensaje privado.
- asunto: pequeña descripción sobre que va el mensaje privado. Es como el asunto de los correos electrónicos, simplemente tiene una funcionalidad informativa.
- texto: texto del mensaje privado.
- fecha: fecha de escritura del mensaje privado.
- destinatario: Identificador del usuario al que va destinado el mensaje privado.
- respuesta: campo que indica si es un mensaje privado de respuesta o no.
- id_resp: id del mensaje al que responde o de los mensajes que forman la conversación.
- leído: campo que indica si el mensaje ha sido abierto o no.
- borrado_aut: campo que indica si el mensaje privado ha sido borrado por el autor o no.
- borrado_dest: campo que indica si el mensaje privado ha sido borrado por el destinatario o no.

Métodos

Los métodos de esta clase son los set y get para cada atributo.

Sigue.java

Contiene a dos usuarios. Representa el seguimiento de un usuario sobre otro.

Atributos

- n_usu1: identificador del usuario que sigue.
- n_usu2: identificador del usuario seguido.
- bloqueado: campo que indica si el usuario 2 ha bloqueado al usuario 1. La función de este atributo es mantener la consistencia de los bloqueos del sistema.

Métodos

Los métodos de esta clase son los set y get para cada atributo.

Bloqueado.java

Representa el bloqueo de un usuario sobre otro.

Atributos

- bloqueado: identificador del usuario bloqueado.
- bloqueante: identificador del usuario bloqueante.

Métodos

Los métodos de esta clase son los set y get para cada atributo.

BusquedaMensajes.java

Clase que sirve para buscar los mensajes, ya sea mediante un identificador, o mediante el filtro de búsqueda. Recoge todos los datos de búsqueda obtenidos del filtro de búsqueda.

Atributos

- idmsg: identificador del mensaje a buscar.
- seguidos: cadena con todos los seguidos de un usuario. Se utiliza cuando se quieren obtener los mensajes de todos los seguidos.
- usuario: identificador del usuario del que se quieren obtener los mensajes.
- fecha1: fecha de la que se quieren obtener los mensajes.
- fecha2: segunda fecha que formará un intervalo con la primera para obtener los mensajes de dicho intervalo.
- opción: campo que indica si se hará una búsqueda de un día o un intervalo de fechas.
- conectado: identificador del usuario que está realizando la búsqueda.
- índice: número que índica a partir de que mensajes hay que obtener, dependiendo de la hoja del tablón que este activa ya que el tablón solo muestra 10 mensajes en cada página.
- grupo: identificador de grupo del que se quieren obtener los menajes.

Métodos

Los métodos de esta clase son los set y get para cada atributo.

A continuación se exponen las clases de la capa Dominio con sus atributos y métodos.

Usuario.java

```
+n_completo: String
+n_usuario: String
+pass: String
+email: String
+fecha_act: String
+f_ult_tablon: String
+num_mensajes: Integer
+getN_completo()
+setN_completo(n_completo:String)
+getN_usuario()
+setN_usuario(n_usuario:String)
+getPass()
+setPass(pass:String)
+getEmail()
+setEmail(email:String)
+getFecha_act()
+setFecha_act(fecha_act:String)
+getF_ult_tablon()
+setF_ult_tablon(f_ult_tablon:String)
+getNum_mensajes()
+setNum_mensajes(num_mensajes:Integer)
```

Ilustración 15. Clase Usuario.java

Mensaje.java

```
+id: String
+usuario: String
+texto: String
+fecha: String
+reescrito: String
+autor: String
+fecha_re: String
+respuesta: String
+num_respuesta: String
+destinatario: String
+grupo: String
+getId()
+setId(id:String)
+getUsuario()
+setUsuario(usuario:String)
+getTexto()
+setTexto(texto:String)
+getFecha()
+setfecha(fecha:String)
+getReescrito()
+setReescrito(reescrito:String)
+getAutor()
+setAutor(autor:String)
+getFecha_re()
+setFecha_re(fecha_re:String)
+getRespuesta()
+setRespuesta(respuesta:String)
+getNum_respuesta()
+setNum_respuesta(num_respuesta:String)
+getDestinatario()
+setDestinatario(destinatario:String)
+getGrupo()
+setGrupo(grupo:String)
```

Ilustración 17. Clase Mensaje.java

Sigue.java

```
+n_usu1: String
+n_usu2: String
+bloqueado: String
+getN_usu1()
+setN_usu1(n_usu1:String)
+getN_usu2()
+setN_usu2(n_usu2:String)
+getBloqueado()
+setBloqueado(bloqueado:String)
```

Ilustración 14. Clase Sigue.java

Privado.java

```
+id: String
+autor: String
+asunto: String
+texto: String
+fecha: String
+destinatario: String
+respuesta: String
+id_resp: String
+leido: String
+borrado_aut: String
+borrado_dest: String
+getId()
+setId(id:String)
+getAutor()
+setAutor(autor:String)
+getAsunto()
+setAsunto(asunto:String)
+getTexto()
+setTexto(texto:String)
+getFecha()
+setFecha(fecha:String)
+getDestinatario()
+setDestinatario(destinatario:String)
+getRespuesta()
+setRespuesta(respuesta:String)
+getId_resp()
+setId_resp(id_resp:String)
+getLeido()
+setLeido(leido:String)
+getBorrado_aut()
+setBorrado_aut(borrado_aut:String)
+getBorrado_dest()
+setBorrado_dest(borrado_dest:String)
```

Ilustración 16. Clase Privado.java

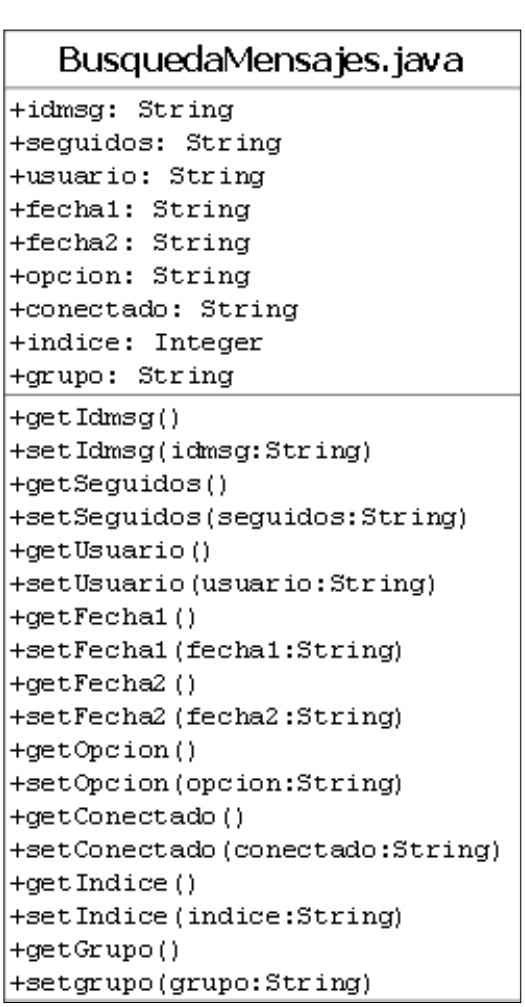


Ilustración 19. Clase BusquedaMensajes.java

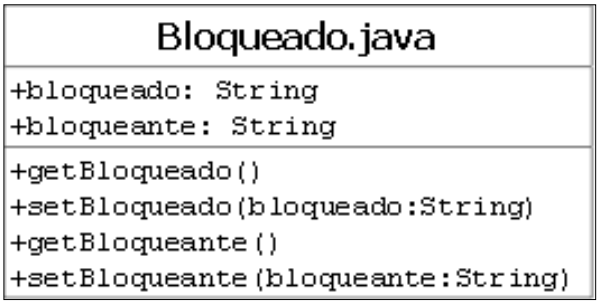


Ilustración 18. Clase Bloqueado.java

5.3.1.3 Capa de datos (DAO)

Las clases de esta capa son las que realizan los accesos a los datos, ya sea la base de datos o el directorio. Los métodos de la clase servicio.java hacen uso de estas clases para acceder a los datos.

La conexión a la base de datos está configurada para obtenerse en el constructor de cada una de las clases, de esta forma el proceso se hace transparente a la hora de desarrollar. Además, la mayoría de las sentencias a la base de datos están fuera de la aplicación en un fichero properties, consiguiendo así evitar tocar el código si se tuviera algún problema con ellas o si se quisieran cambiar en un futuro.

Al separar el acceso a los datos de la capa de servicio se consigue una mayor modularidad y permite una mayor facilidad a la hora de desarrollar y resolver los problemas que pudieran surgir al tener estos mejor localizados.

A continuación expondré y explicare brevemente las clases existentes en esta capa:

UsuarioDao.java

Esta clase realiza la conexión a la base de datos e implementa las sentencias sobre la tabla usuario.

Atributos

- (ResultSet) resultado: contendrá todas las filas que satisfagan las condiciones de las sentencias SQL y proporcionará el acceso a los datos de estas filas.
- (PreparedStatement) stm: permite enviar sentencias SQL a la base de datos. La característica esencial es que permite usando la misma sentencia enviar varios datos diferentes.
- (Connection) DriverConnection: un objeto 'connection' que permite obtener la conexión a la base de datos a través del DataSource.
- (Properties) prop: objeto 'properties' que permitirá cargar un fichero externo de pares de valores donde se almacenarán la mayoría de sentencias SQL.

Métodos

- c_primerLogin: realiza la query que comprueba si un usuario se ha conectado alguna vez al sistema o es la primera vez.
- registrar_datosLDAP_usuario: realiza la query que introduce en la tabla de usuarios los datos obtenidos del directorio para que el sistema funcione correctamente.
- actualizarFecha: realiza la query que actualiza la fecha de última actividad de un usuario.
- incrementar_mensajes: realiza la query que incrementa el campo número de mensajes de un usuario.
- decrementar_mensajes: realiza la query que decrementa el campo número de mensajes de un usuario.

- buscarEmail: realiza la query que obtiene la información de un usuario dado su email.
- buscarNombre: realiza la query que obtiene la información de un usuario dado su identificador.
- obtenerFechaTablon: realiza la query que obtiene la fecha de última actualización del tablón de mensajes de un usuario.
- obtenerFechasActualizadas: realiza la query que obtiene las fechas de última actividad de un grupo de usuarios seguidos.
- actualizarFechaTablon: realiza la query que actualiza la fecha de última actualización del tablón de mensajes de un usuario.

MensajeDao.java

Esta clase realiza la conexión a la base de datos e implementa las peticiones sobre la tabla mensaje.

Atributos

- (ResultSet) resultado: contendrá todas las filas que satisfagan las condiciones de las sentencias SQL y proporcionará el acceso a los datos de estas filas.
- (PreparedStatement) stm: permite enviar sentencias SQL a la base de datos. La característica esencial es que permite usando la misma sentencia enviar varios datos diferentes.
- (Connection) DriverConnection: un objeto 'connection' que permite obtener la conexión a la base de datos a través del DataSource.
- (Properties) prop: objeto 'properties' que permitirá cargar un fichero externo de pares de valores donde se almacenarán la mayoría de querys.

Métodos

- registrar_msg: realiza la query que registra un nuevo mensaje en la tabla de mensajes.
- eliminarMensaje: realiza la query que elimina un mensaje de la tabla de mensajes.
- obtenerMensajes: realiza la query que obtiene los mensajes a mostrar para un usuario.
- contarObtenerMensajes: realiza la query que cuenta los resultados que devuelve la función obtenerMensajes, necesario para realizar correctamente la paginación del tablón de mensajes.
- obtenerContMsg: realiza la query que devuelve la información (autor, texto, fecha) de un mensaje concreto.
- obtenerGrupoMensaje: realiza la query que obtiene el grupo al que pertenece un mensaje.
- buscarIdRespondido: realiza la query que obtiene el identificador del mensaje al que responde otro pasado como parámetro.
- obtenerMensajesGrupo: realiza la query que obtiene los mensajes a mostrar de un grupo determinado.

- contarObtenerMensajesGrupo: realiza la query que cuenta los resultados que devuelve la función obtenerMensajesGrupo necesario para realizar correctamente la paginación del tablón de mensajes.
- obtenerRespuestas: realiza la query que obtiene los mensajes de respuesta de otro pasado como parámetro.
- contarObtenerRespuestas: realiza la query que cuenta los resultados que devuelve la función obtenerRespuestas necesario para realizar correctamente la paginación del tablón de respuestas.

PrivadoDao.java

Esta clase realiza la conexión a la base de datos e implementa las querys sobre la tabla privado.

Atributos

- (ResultSet) resultado: contendrá todas las filas que satisfagan las condiciones de las sentencias SQL y proporcionará el acceso a los datos de estas filas.
- (PreparedStatement) stm: permite enviar sentencias SQL a la base de datos. La característica esencial es que permite usando la misma sentencia enviar varios datos diferentes.
- (Connection) DriverConnection: un objeto 'connection' que permite obtener la conexión a la base de datos a través del DataSource.
- (Properties) prop: objeto 'properties' que permitirá cargar un fichero externo de pares de valores donde se almacenarán la mayoría de querys.

Métodos

- obtenerIdRespPriv: realiza la query que obtiene los identificadores a los que responde otro pasado como parámetro. Si es parte de una conversación obtendrá todos los identificadores pertenecientes a esta.
- registrar_privado: realiza la query que introduce un nuevo mensaje privado con los datos introducidos por el usuario.
- obtenerPrivados: realiza la query que obtiene los mensajes privados a mostrar para un usuario. Los mensajes pueden ser nuevos, leídos o enviados.
- obtenerContPrivados: realiza la query que obtiene el contenido de los mensajes privados pasados como parámetros.
- actualizarLeidoPrivado: realiza la query que modifica el estado de un mensaje privado como leído.
- obtener_borrados_priv: realiza la query que obtiene los campos de borrado de un mensaje privado. Estos campos indican quien ha borrado el privado, el autor, el destinatario o si no ha sido borrado.
- modificarBorradoPriv: realiza la query que marca como borrado un mensaje privado. En el caso de que ni autor ni destinatario hayan borrado el mensaje se modificará el campo correspondiente.

- eliminarPrivado: realiza la query que elimina un mensaje privado de la tabla de privados cuando ambos, autor y destinatario, han borrado el mensaje.

SigueDao.java

Esta clase realiza la conexión a la base de datos e implementa las querys sobre la tabla sigue.

Atributos

- (ResultSet) resultado: contendrá todas las filas que satisfagan las condiciones de las sentencias SQL y proporcionará el acceso a los datos de estas filas.
- (PreparedStatement) stm: permite enviar sentencias SQL a la base de datos. La característica esencial es que permite usando la misma sentencia enviar varios datos diferentes.
- (Connection) DriverConnection: un objeto 'connection' que permite obtener la conexión a la base de datos a través del DataSource.
- (Properties) prop: objeto 'properties' que permitirá cargar un fichero externo de pares de valores donde se almacenarán la mayoría de querys.

Métodos

- registrar_datosLDAP_sigue: realiza la query que inserta los datos obtenidos del directorio LDAP al ingresar por primera vez en la aplicación. Inserta la tupla sigue del usuario sobre sí mismo, necesaria para que se muestren sus propios mensajes en el tablón.
- anadir_seguido: realiza la query que añade un nuevo seguido.
- modificar_seguido: realiza la query que modifica el campo bloqueado de la tabla sigue que indica si un usuario ha bloqueado a otro, por lo que invalida esta tupla.
- comprobar_seguido: realiza la query que comprueba si un usuario sigue a otro.
- obtenerSeguidos: realiza la query que obtiene los seguidos de un usuario añadiéndose a sí mismo. La razón para añadirse a sí mismo es para que aparezcan también los mensajes propios en el tablón de mensajes.
- obtenerSeguidosPropiosOrd: realiza la query que obtiene los seguidos de un usuario sin añadirse a sí mismo ordenados por el número de mensajes que hayan escrito.
- obtenerSeguidoresPropiosOrd: realiza la query que obtiene los seguidores de un usuario sin añadirse a si mismo ordenado por el número de mensajes que hayan escrito.
- eliminarSeguido: realiza la query que elimina una tupla de la tabla sigue dados dos usuarios.
- contarSeguidos: realiza la query que cuenta el número de seguidos de un usuario.
- contarSeguidores: realiza la query que cuenta el número de seguidores de un usuario.

BloqueadoDao.java

Esta clase realiza la conexión a la base de datos e implementa las querys sobre la tabla bloqueado.

Atributos

- (ResultSet) resultado: contendrá todas las filas que satisfagan las condiciones de las sentencias SQL y proporcionará el acceso a los datos de estas filas.
- (PreparedStatement) stm: permite enviar sentencias SQL a la base de datos. La característica esencial es que permite usando la misma sentencia enviar varios datos diferentes.
- (Connection) DriverConnection: un objeto 'connection' que permite obtener la conexión a la base de datos a través del DataSource.
- (Properties) prop: objeto 'properties' que permitirá cargar un fichero externo de pares de valores donde se almacenarán la mayoría de querys.

Métodos

- obtener_bloqueados: realiza la query que comprueba si existe una tupla concreta de bloqueado, es decir comprueba si un usuario a bloqueado a otro o no.
- anadir_bloqueado: realiza la query que añade un nuevo bloqueado a la tabla de bloqueados dados dos usuarios.
- eliminar_bloqueado: realiza la query que elimina una tupla de bloqueados cuando un usuario desbloquea a otro.

LdapDao.java

Esta clase realiza la conexión al directorio y realiza las búsquedas de datos sobre él.

Atributos

- (Hashtable) entorno: con este atributo se crea el entorno que permitirá conectarse al directorio y realizar las búsquedas.

Métodos

- obtenerUsuariosGrupo: obtiene del directorio los usuarios que pertenecen a un grupo concreto.
- buscarDN: busca y obtiene del directorio el DN (Nombre distinguido) de un usuario utilizando un campo único.
- autenticarDN: con el DN del usuario obtenido en la función anterior, autentica el usuario sobre el directorio dando o no acceso si la clave de usuario es correcta.
- buscarAtributos: obtiene el email de un usuario del directorio.
- obtenerGrupos: obtiene del directorio los grupos a los que pertenece un usuario concreto.

Clases de la capa de datos.

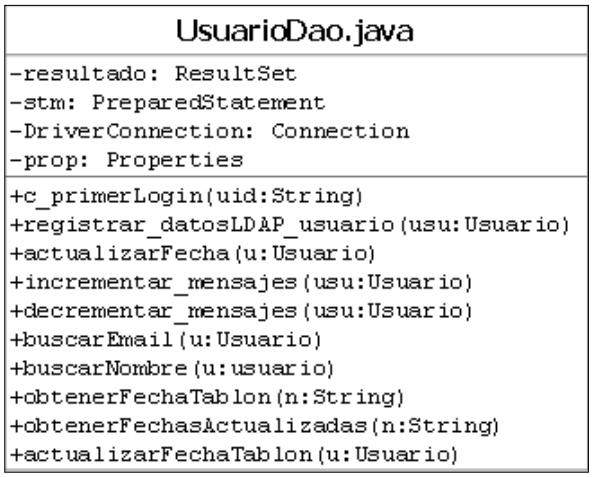


Ilustración 24. Clase UsuarioDao.java

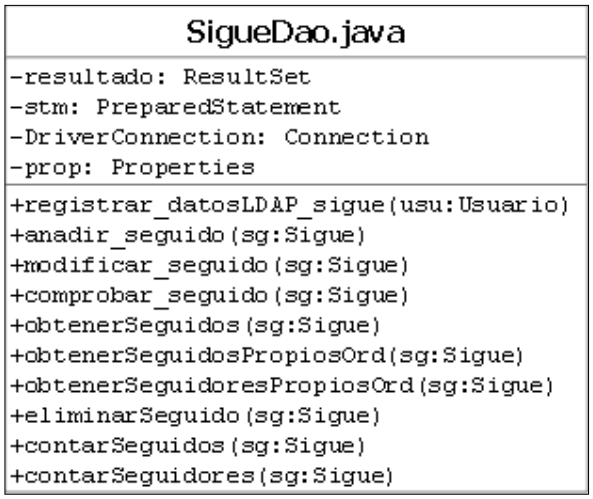


Ilustración 23. Clase SigueDao.java

Ilustración 22. Clase MensajeDao.java

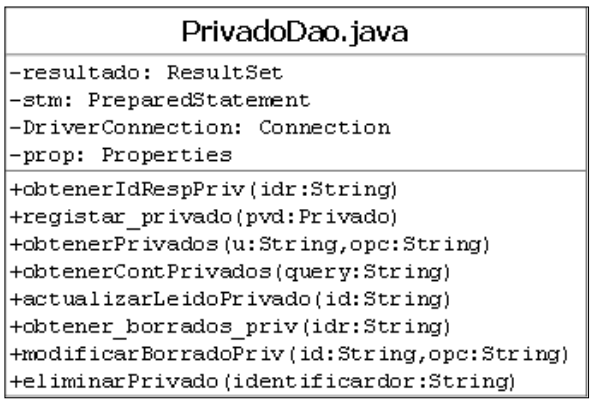


Ilustración 21. Clase PrivadoDao.java

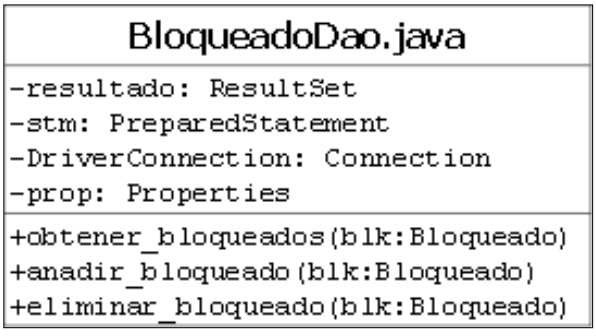


Ilustración 25. Clase BloqueadoDao.java

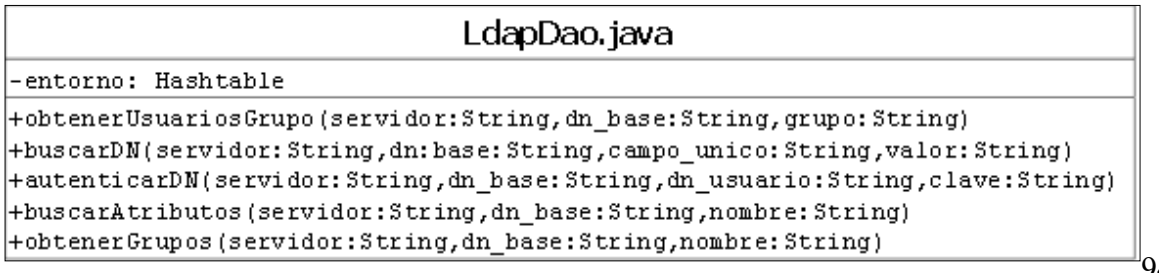


Ilustración 20. Clase LdapDao.java

5.3.2 Diagrama de estructura.

A continuación expondré el diagrama estático donde se puede observar la relación entre las clases que forman la aplicación.

En el diagrama se puede observar los paquetes que contienen las clases expuestas en los apartados anteriores. Dentro de cada paquete se puede observar la representación de las clases. No se muestran los atributos y métodos para que el esquema no quede demasiado sobrecargado.

Aunque la conexión es entre paquetes lo que representa es que la clase servicio se conecta con cada una de las clases de los paquetes. Cada clase de los paquetes Dominio y DAO no tienen relación entre sí mismas. Si algún método de la clase servicio tiene que hacer uso de varios métodos de distintas clases, lo hará de forma secuencial.

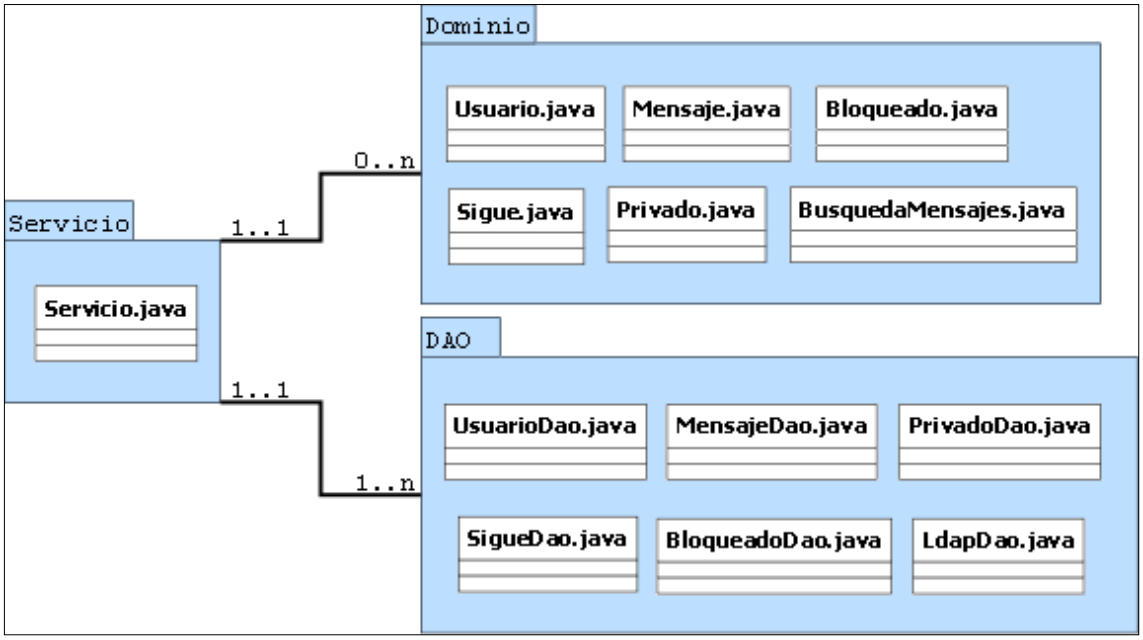


Ilustración 26. Diagrama de clases.

Los diagramas de las clases con sus atributos y métodos se exponen en los apartados vistos previamente, donde se explican cada una de ellas.

Una vez visto el diseño estático de la aplicación, voy a explicar el diseño dinámico, donde se podrán observar los diagramas de secuencia para cada caso de uso y así entender mejor cómo funciona la aplicación y como los usuarios interactúan con ella.

5.4 Diseño dinámico.

En este apartado voy a exponer los diagramas de secuencia de las principales funcionalidades de la aplicación. Habrá un diagrama por cada caso de uso. Mientras que los casos de uso dan una idea general de cuál es el funcionamiento de la aplicación, los diagramas de secuencia contienen detalles de implementación por lo que, junto con el diagrama estático, es una buena forma de describir como se han implementado las diferentes funcionalidades, ya que modelan la interacción entre objetos a través del tiempo, mostrando las clases y métodos que se usan para cada operación.

Antes de exponer los diagramas quiero explicar brevemente algunos aspectos sobre ellos. Sobre cada mensaje aparece el método de la clase destino que se ejecuta con los parámetros necesarios. Los métodos que devuelven algo, tienen las respuestas indicadas como una flecha con línea discontinua y encima un mensaje descriptivo de que es lo que devuelve y de qué tipo (entre corchetes). Cuando en la ejecución hay un bucle o algún condicional, se explica con un comentario que indica que mensajes entran dentro de estos.

A continuación paso a exponer dichos diagramas.

5.4.1 Diagramas de secuencia.

5.4.1.1 Autenticación

El usuario introduce su identificador y clave para conectarse al sistema. La capa servicio se encarga de buscar el DN de dicho usuario en el directorio. Cuando lo obtiene autentica al usuario con la clave. Una vez que se ha validado el usuario, comprueba si es la primera vez que el usuario entra en la aplicación y si es así, obtiene los atributos necesarios del directorio y los introduce en la base de datos de la aplicación.

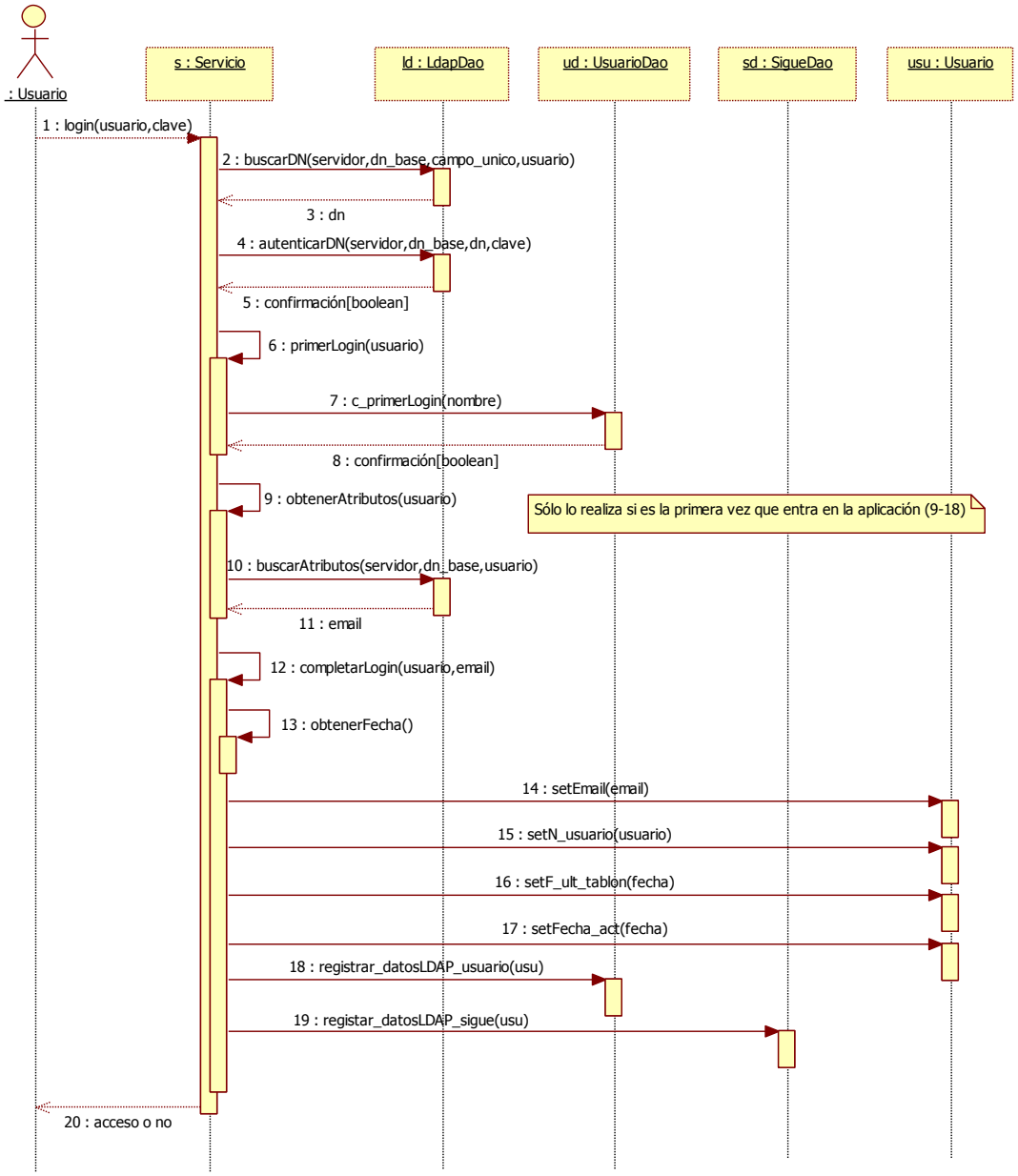


Ilustración 27. Diagrama se secuencia - Autenticación.

5.4.1.2 Leer mensajes.

Primero se obtienen los usuarios a los que se sigue para, a continuación, buscar los mensajes correspondientes. Para hacer estas búsquedas la clase servicio hace uso de las clases de la capa dominio. Una vez que se han obtenido los mensajes se transforman los datos para enviarlos al lado cliente donde se presentarán en la interfaz de usuario.

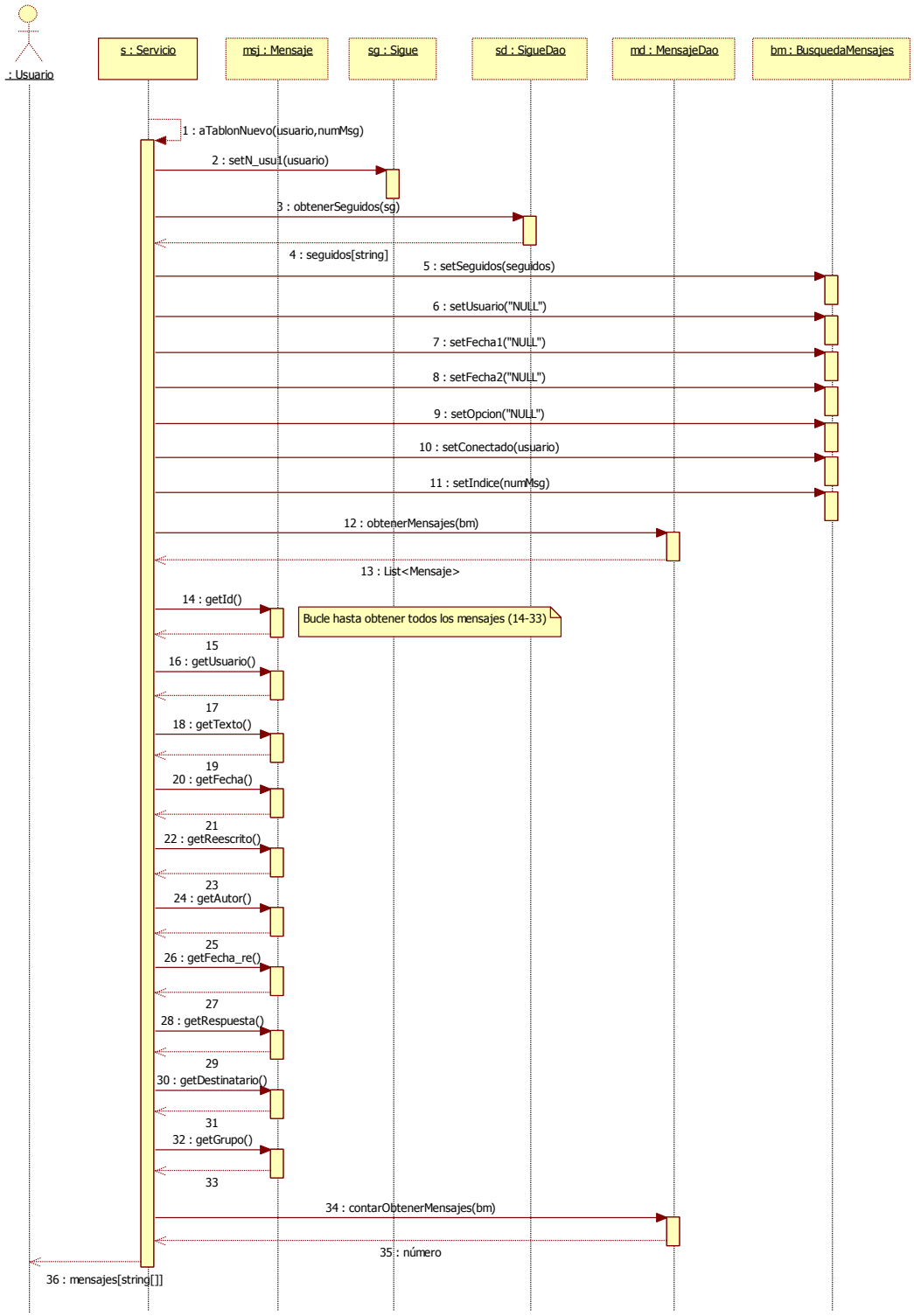


Ilustración 28. Diagrama de secuencia - Leer mensajes.

5.4.1.3 Filtrar mensajes.

No expongo el diagrama ya que es muy similar al anterior, pero cambiando los parámetros de búsqueda de los mensajes 5 a 11 según como el usuario haya interactuado con la aplicación.

5.4.1.4 Modificar página del tablón.

El diagrama es igual que el de Leer mensajes pero cambiando el parámetro del mensaje 11.

5.4.1.5 Buscar usuario.

El usuario envía el tipo de búsqueda (nombre o email) y el texto de búsqueda. Con estos datos se busca en la base de datos algún usuario que corresponda con los datos enviados y si se encuentra se devuelven los datos para poder añadirlo como seguido.

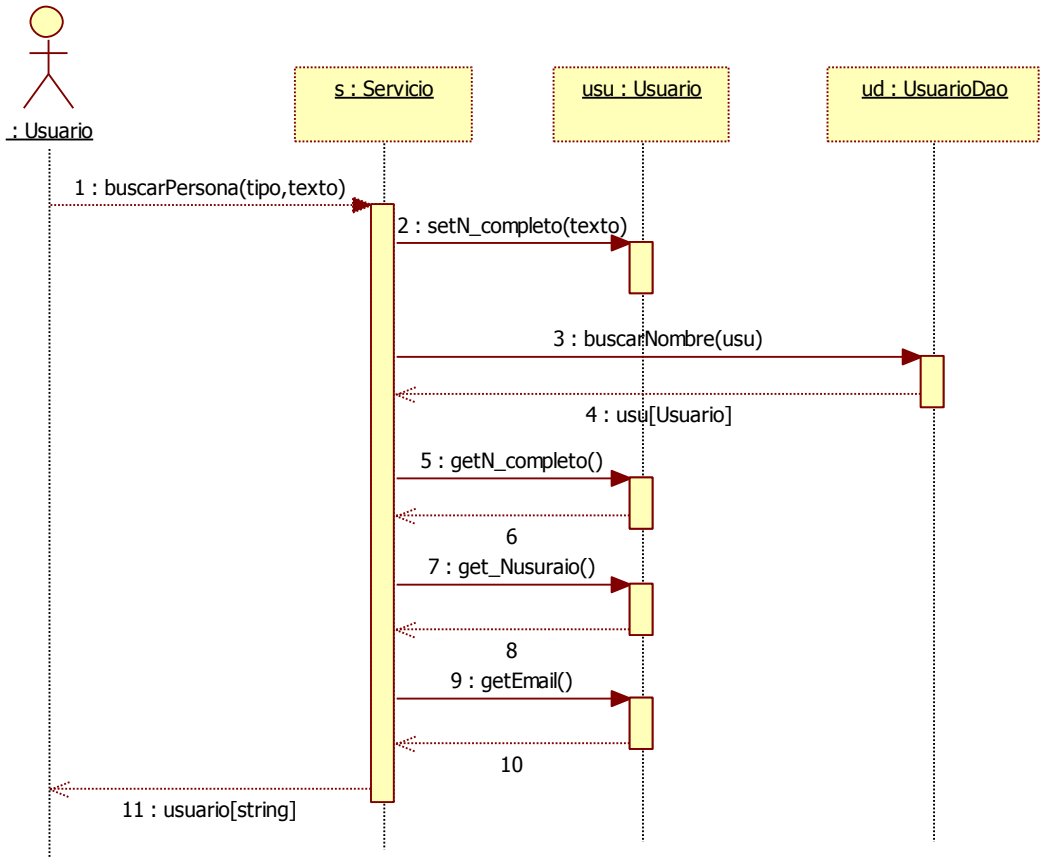


Ilustración 29. Diagrama de secuencia - Buscar usuario.

5.4.1.6 Seguir usuario.

Una vez el usuario ha obtenido un resultado, hace clic sobre el botón de seguir en la interfaz de usuario y esto envía los datos de los usuarios involucrados. Lo primero es buscar si el usuario está bloqueado o no para luego introducir la tupla del seguimiento actualizado con el campo de bloqueo obtenido.

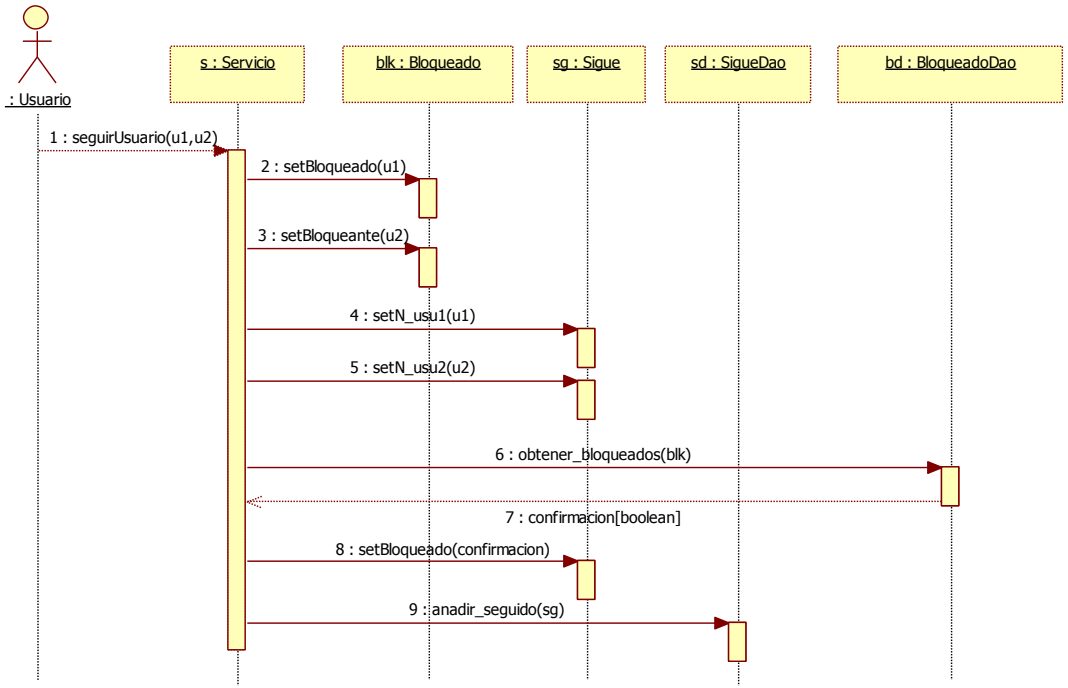


Ilustración 30. Diagrama de secuencia - Seguir seguido.

5.4.1.7 No seguir usuario.

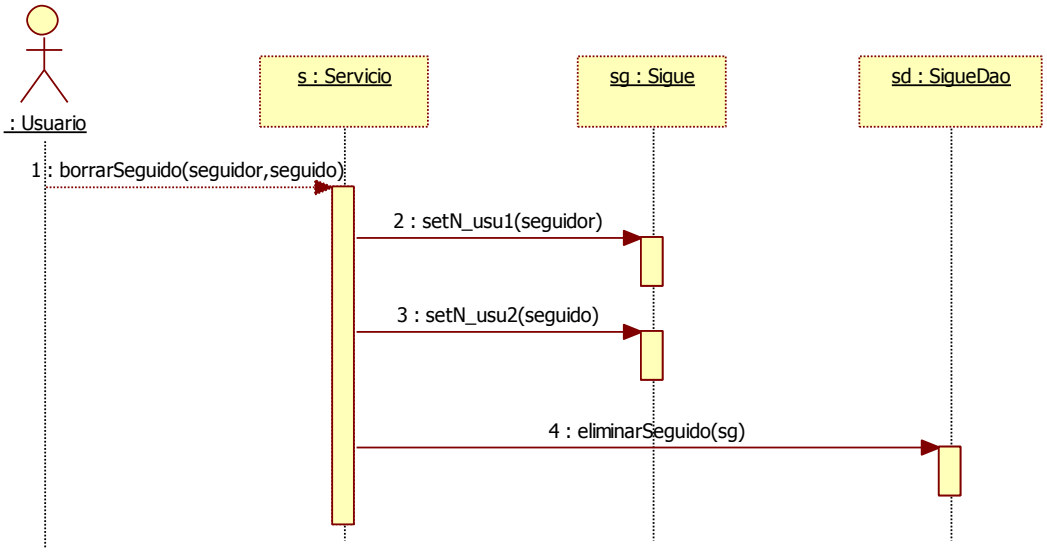


Ilustración 31. Diagrama de secuencia - No seguir usuario.

5.4.1.8 Bloquear usuario.

El usuario inicia la acción de bloqueo pinchando sobre el botón correspondiente de la interfaz de usuario. Se introduce la tupla de bloqueo dentro de su tabla correspondiente y se actualiza la tupla de la tabla sigue para indicar que está bloqueado. Finalmente como es una acción que requerirá que se actualice el tablón se actualiza la fecha de actividad del usuario bloqueante.

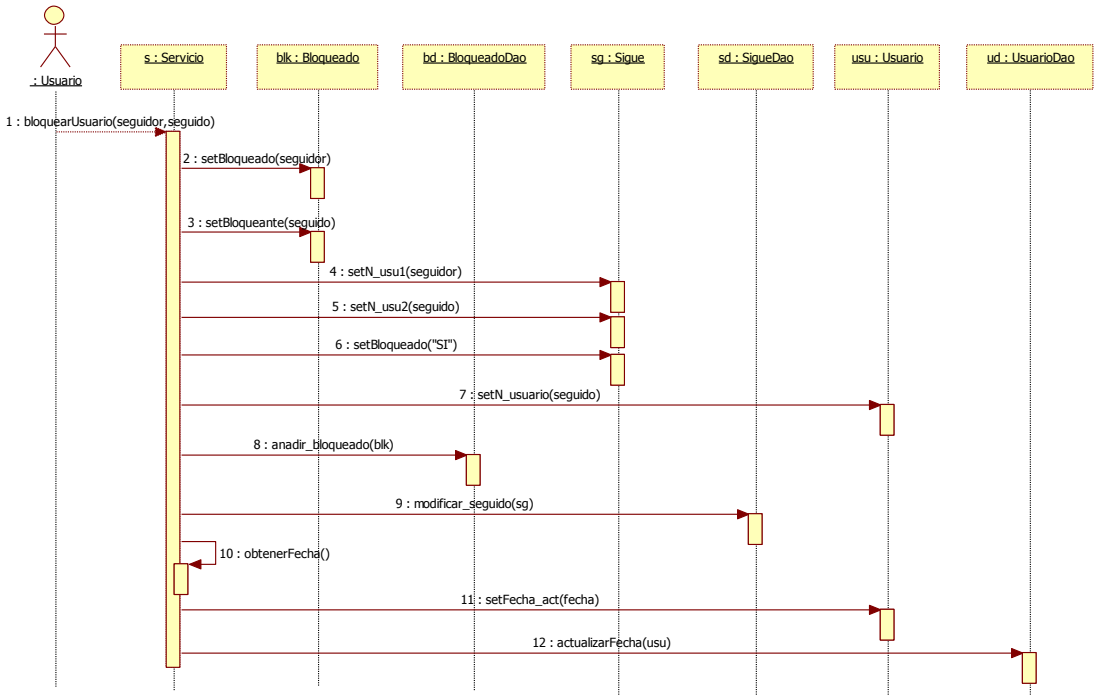


Ilustración 32. Diagrama de secuencia - Bloquear usuario.

5.4.1.9 Desbloquear usuario.

El usuario picha sobre el botón de desbloquear usuario y se envían los dos usuarios implicados. Primero se borra la tupla de la tabla bloqueado con los dos usuarios. A continuación se modifica el campo bloqueado de la tupla de la tabla sigue. Finalmente se actualiza la fecha de actividad de usuario.

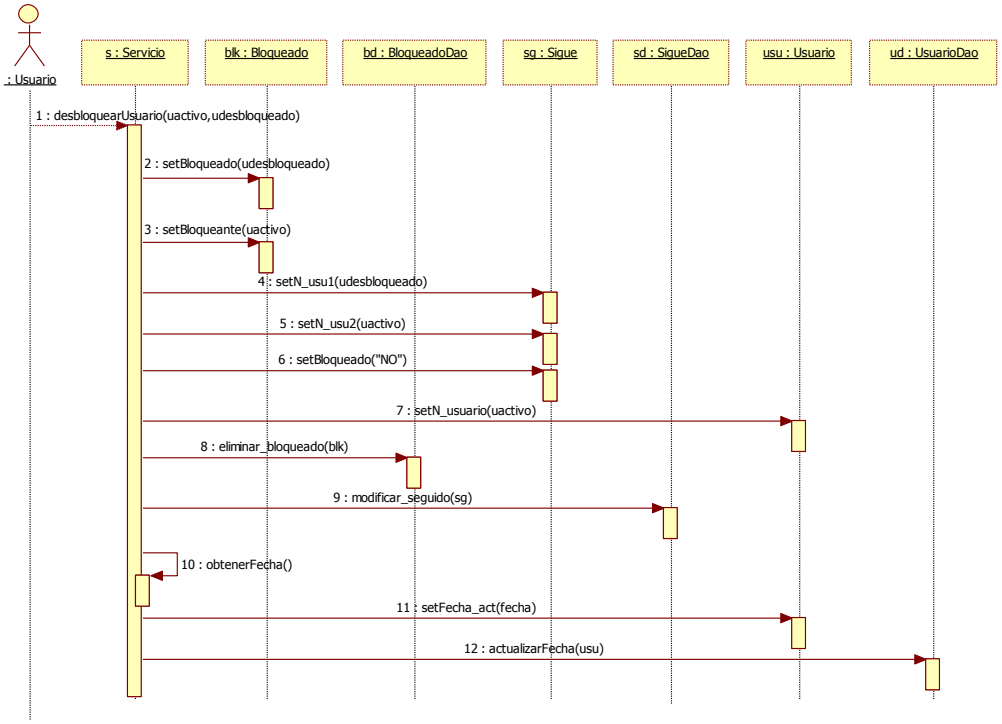


Ilustración 33. Diagrama de secuencia - Desbloquear usuario

5.4.1.10 Ver seguidos.

Primero se obtienen los usuarios seguidos del usuario que está realizando la búsqueda, ordenados por número de mensajes. Seguidamente se obtienen los bloqueos de dichos usuarios para comprobar si alguno nos ha bloqueado. Finalmente se devuelven los resultados para mostrarlos en la interfaz.

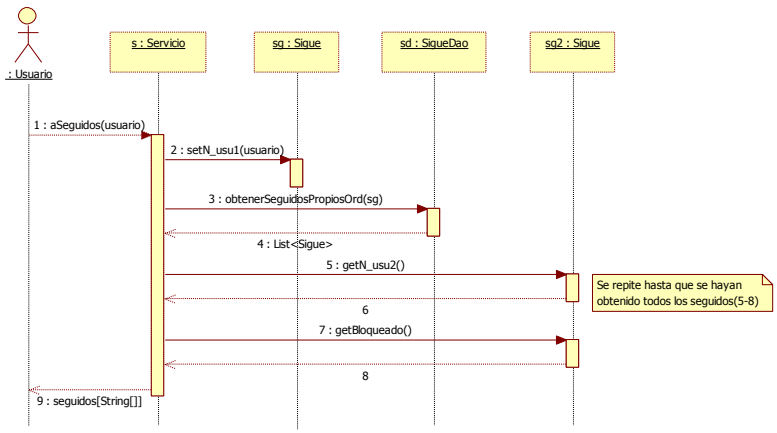


Ilustración 34. Diagrama de secuencia - Ver seguidos.

5.4.1.11 Ver seguidores.

Es igual que ver seguidos pero realizando la búsqueda sobre los seguidores.

5.4.1.12 Escribir mensaje.

Se rellena un objeto mensaje con los datos pasados por el usuario, rellenando sólo los correspondientes a un mensaje normal, estableciendo los campos de reescrito y respuesta a NULL. Después se introduce el mensaje en la base de datos y se actualiza la fecha de actividad del usuario.

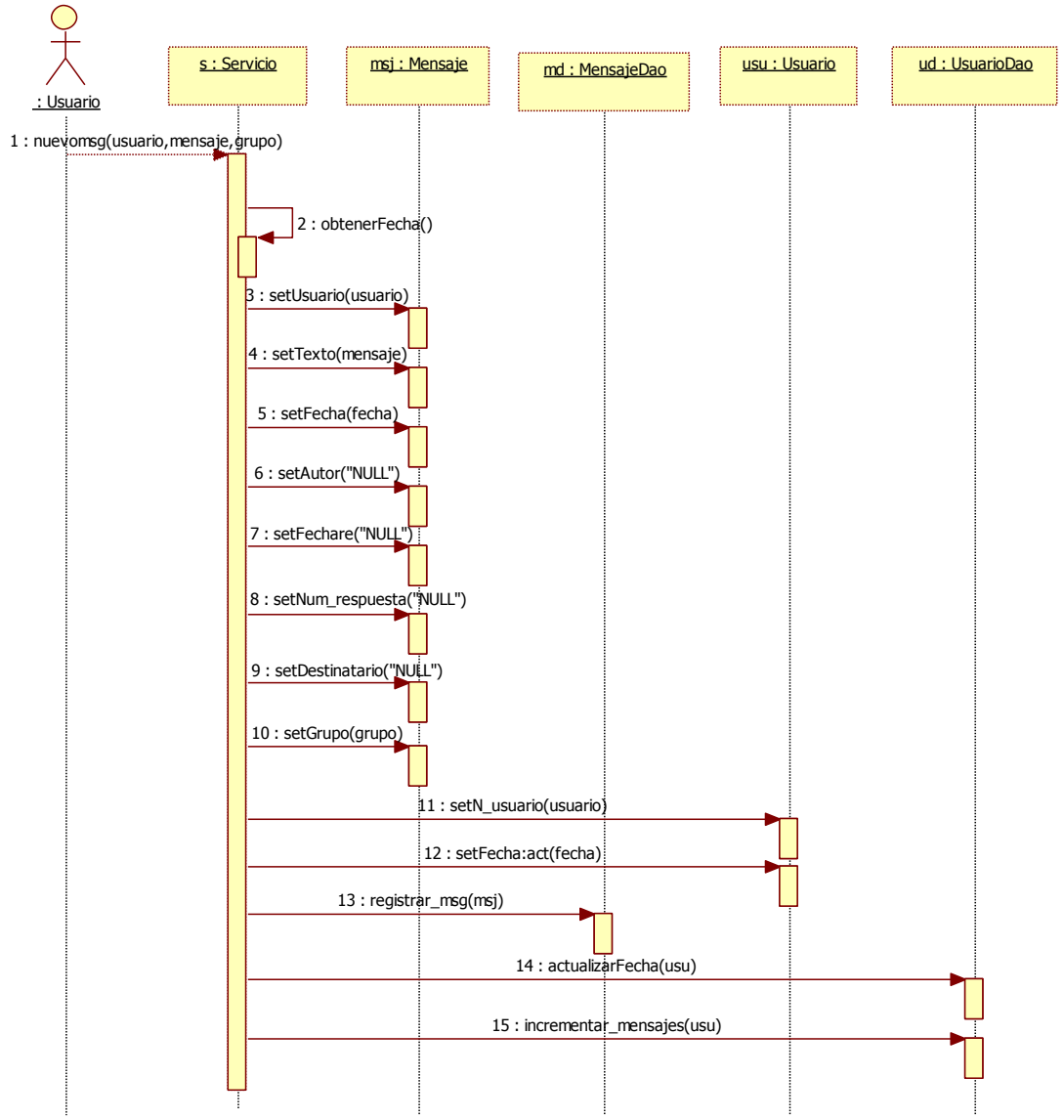


Ilustración 35. Diagrama de secuencia - Escribir mensaje.

5.4.1.13 Borrar mensaje.

Se borra el mensaje con el identificador pasado como parámetro y se actualiza la fecha de actividad del usuario.

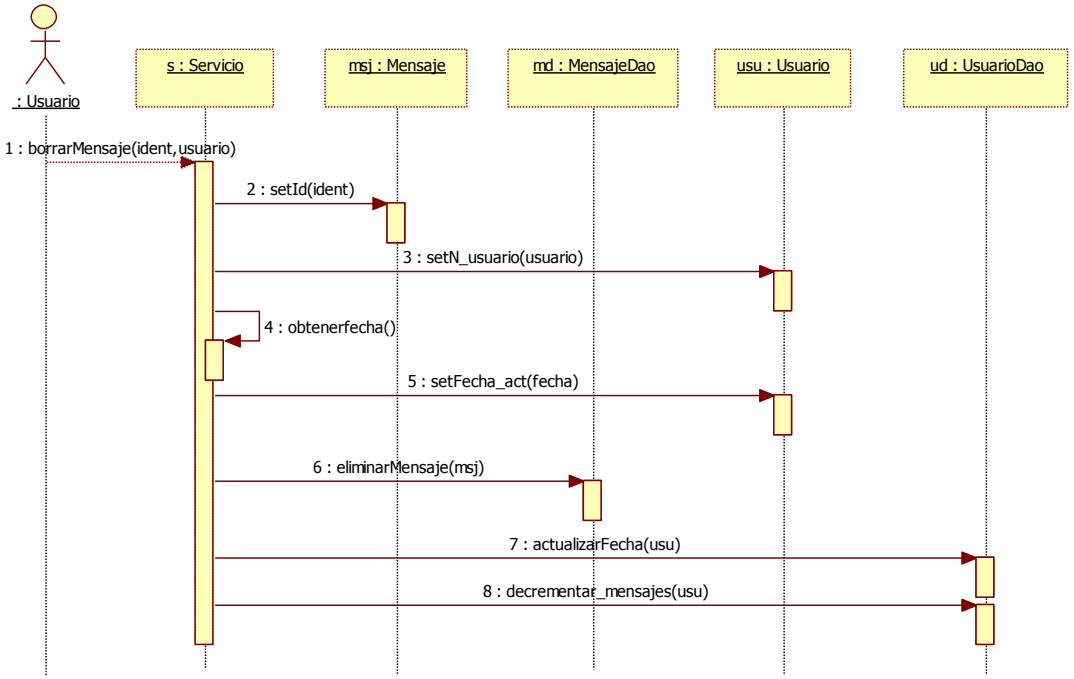


Ilustración 36. Diagrama de secuencia - Borrar mensaje.

5.4.1.14 Reescribir mensaje.

Lo primero que se hace es obtener la información y el contenido del mensaje que se va a reescribir. Sobre el mensaje devuelto se introducen los datos del usuario que va a reescribir el mensaje, estando ya rellenos los campos correspondientes del mensaje reescrito. También se obtiene el grupo del mensaje original para que el nuevo mensaje reescrito vaya al grupo correspondiente. Finalmente se registra el nuevo mensaje, haciendo uso del objeto mensaje que hemos rellenado y se actualiza la fecha de actividad del usuario.

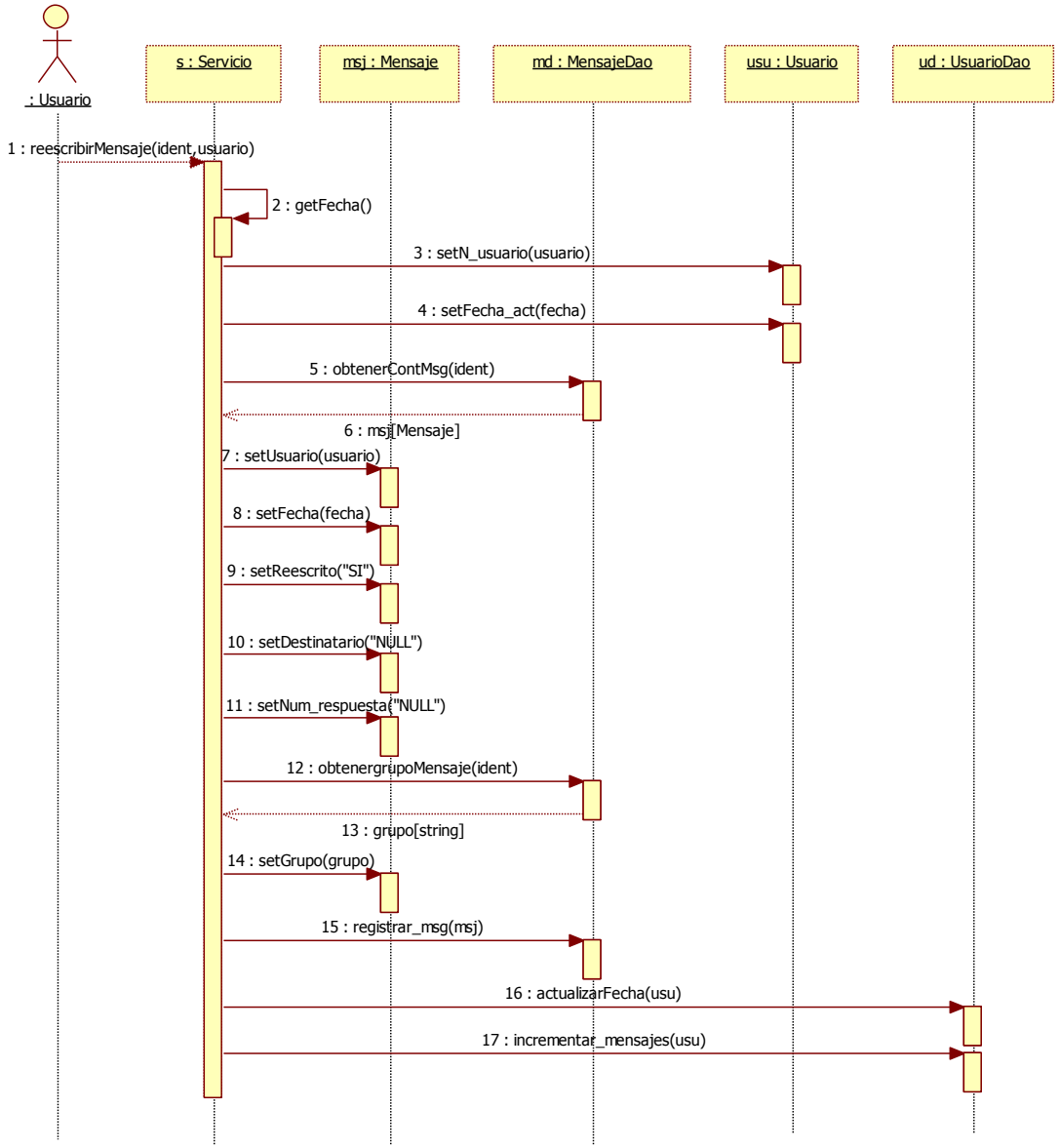


Ilustración 37. Diagrama de secuencia - Reescribir mensaje.

5.4.1.15 Responder mensaje.

Con el identificador del mensaje al que se responde, se busca su autor para luego utilizarlo en la respuesta como destinatario. Se establecen los datos pasados como parámetro junto con el identificador del mensaje respondido, el destinatario y el grupo para insertar el nuevo mensaje de respuesta. Finalmente se actualiza la fecha de actividad del usuario que responde.

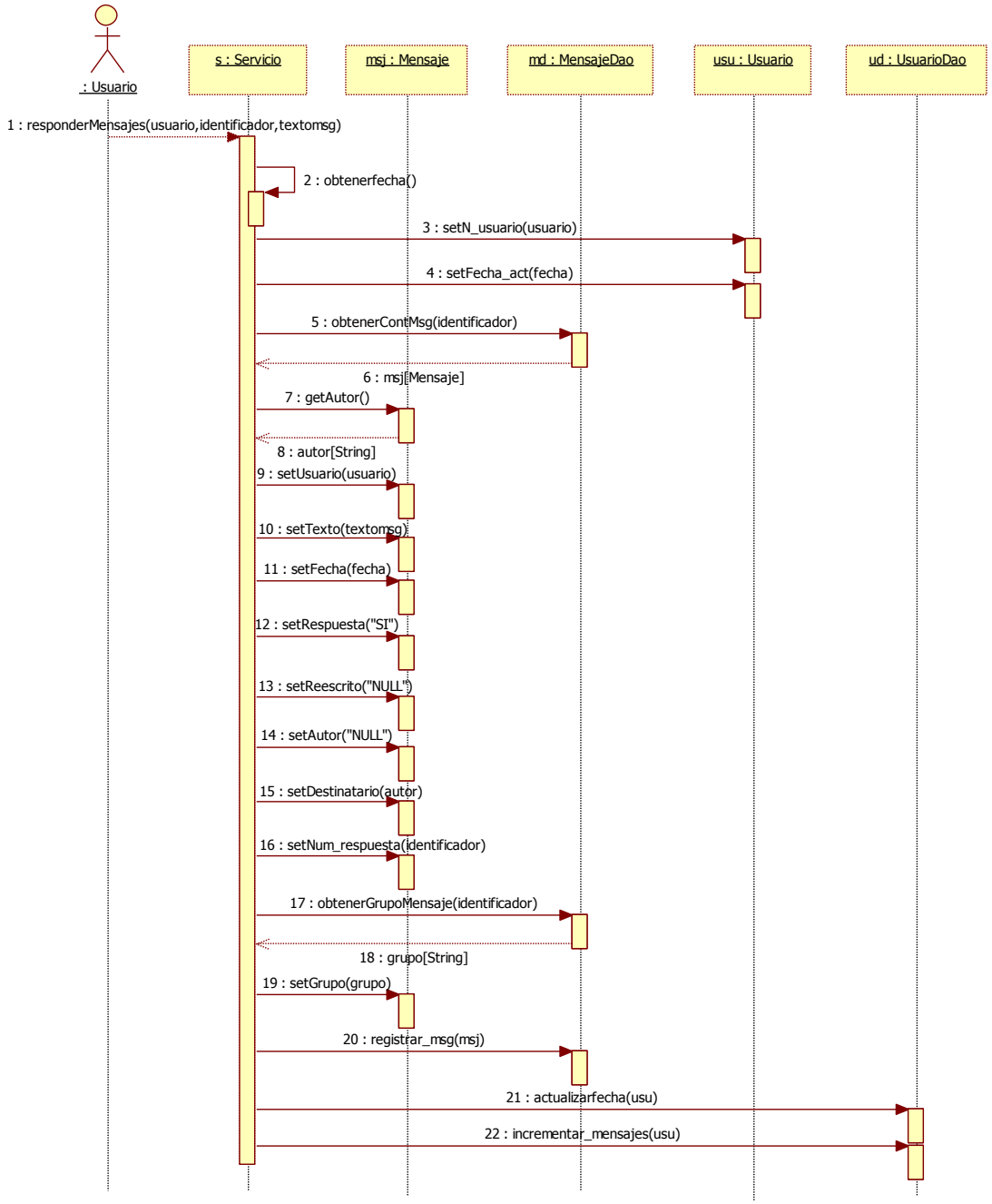


Ilustración 38. Diagrama de secuencia - Responder mensaje.

5.4.1.16 Enviar privado – Responder privado.

Se crea un objeto 'privado' donde se establecen los argumentos pasados para insertar el privado. En el caso de que sea un mensaje privado normal la jerarquía de mensajes de respuesta (mensaje 9) quedará en blanco. Si es un privado de respuesta, se obtiene la jerarquía de privados del mensaje al que se responde, el cual contendrá todos los identificadores de los privados de respuesta de la conversación.

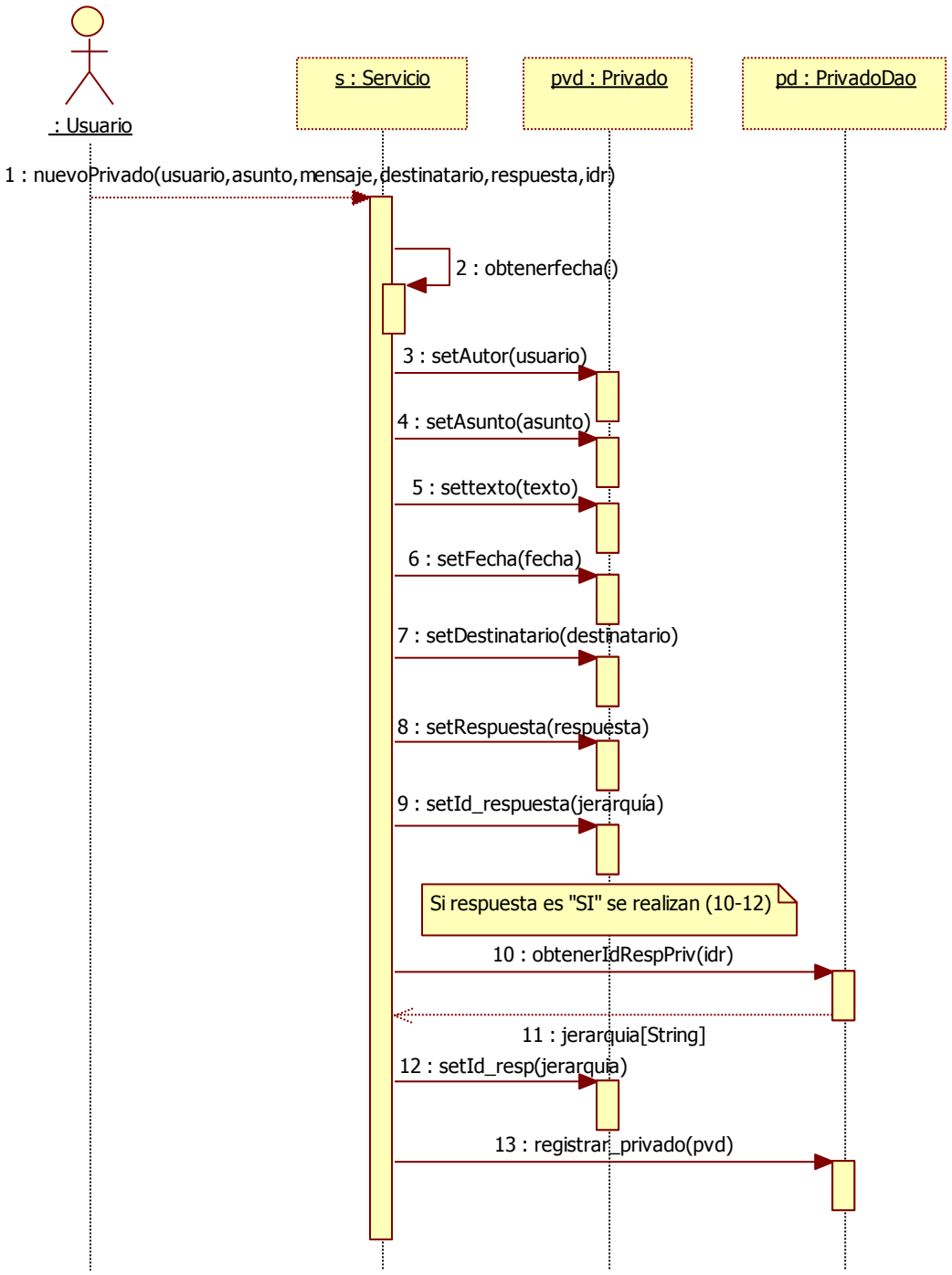


Ilustración 39. Diagrama de secuencia - Enviar privado.

5.4.1.17 Leer privado.

Lo primero que hace es obtener los identificadores de los privados de la conversación del privado que se quiere leer, por si fuera un mensaje que fuera parte de una conversación. Con esos identificadores se obtiene una lista con todos los privados correspondientes. En el caso de que no fuera parte de una conversación, como mínimo obtendría el propio mensaje privado que se quiere leer. Una vez obtenido los mensajes se transforman los datos para pasarlos al lado cliente y que se muestren en la interfaz de usuario.

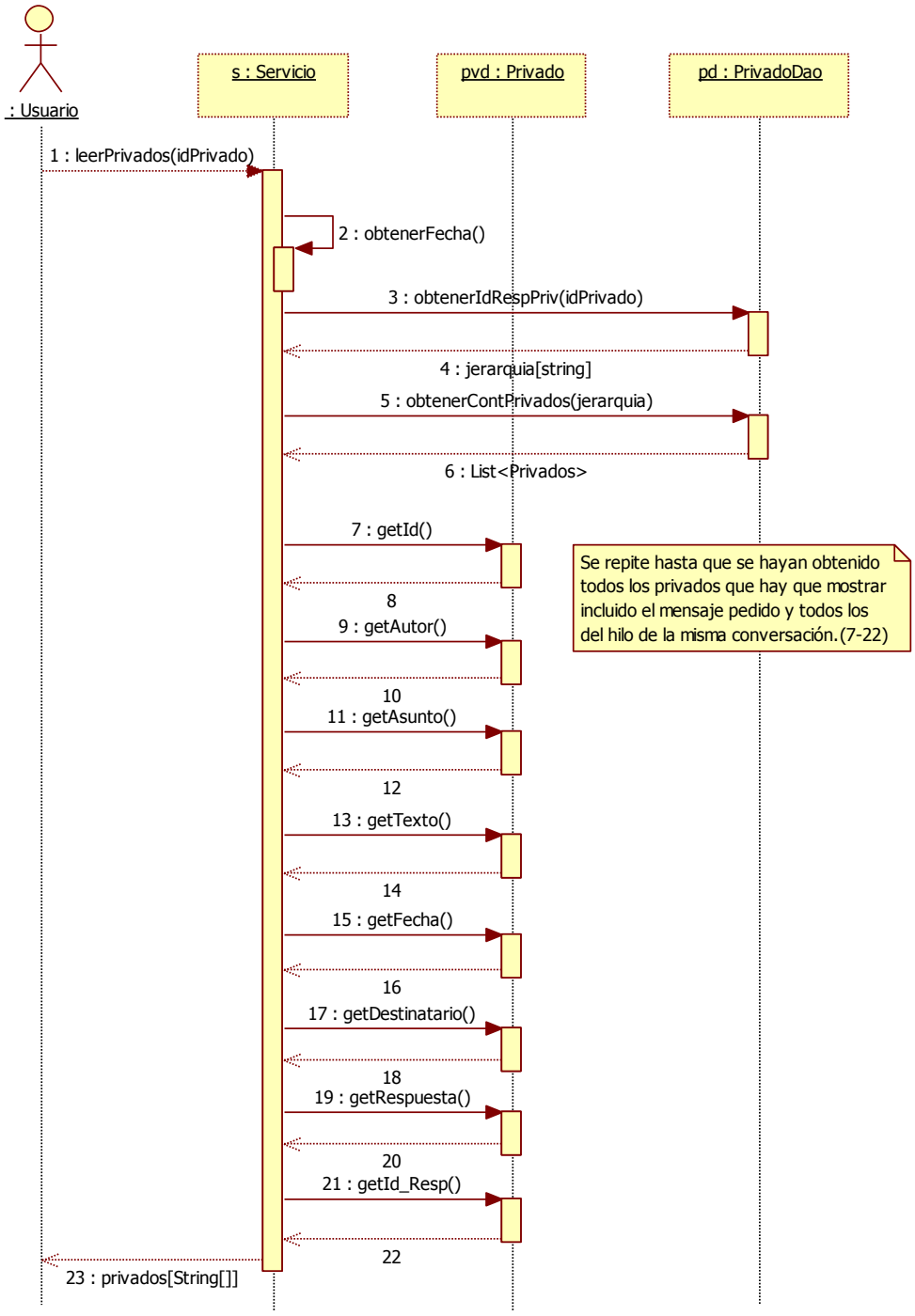


Ilustración 40. Diagrama de secuencia - Leer privado.

5.4.1.18 Borrar privado.

Para borrar un mensaje privado el usuario envía el identificador del mensaje que se quiere borrar y una opción que indica si es el autor o el destinatario del mensaje privado. Primero se obtienen los dos campos de borrado del privado que se quiere borrar. En el caso de que los dos campos sean iguales quiere decir que ninguno ha borrado el mensaje por lo que sólo se modificaría el campo correspondiente a la opción. Si los campos son diferentes quiere decir que uno de los dos ya lo ha borrado por lo que se elimina el mensaje de la base de datos.

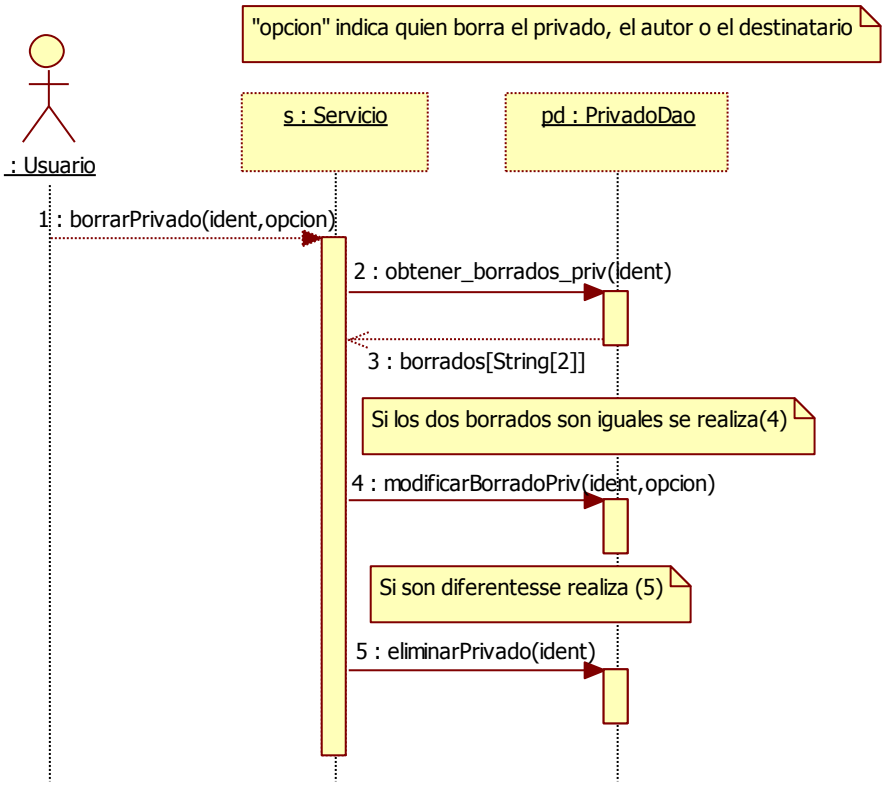


Ilustración 41. Diagrama de secuencia - Borrar privado.

5.5 Diseño y descripción de la base de datos.

En el siguiente apartado se mostrarán el modelo Entidad-Relación que representa la base de datos del sistema con sus tablas, atributos y relaciones, y el grafo relacional, en el cual se transforma el diagrama E-R en tablas donde se podrán observar las claves primarias y externas.

5.5.1 Modelo Entidad-Relación

A continuación se expone el diagrama E-R o modelo E-R. Para realizar el diagrama hay que tener en cuenta que datos son necesarios recoger y que relaciones hay entre ellos en el sistema.

Será necesario recoger los datos correspondientes a los usuarios que se conectan a la aplicación que se agruparán en la tabla 'Usuarios'. Los usuarios podrán escribir mensajes, por lo que habrá que recoger los datos correspondientes a estos, que se almacenarán en la tabla 'Mensajes'. Los usuarios también pueden escribir mensajes privados por lo que se recogerán todos los datos en la tabla 'Privados'. Además, los usuarios tienen la posibilidad de seguirse los unos a los otros por lo que existe una nueva relación que se representa con el rombo 'sigue' que se convertirá en una nueva tabla. De igual forma, los usuarios pueden bloquear a otros usuarios, por lo tanto es necesario otra relación que se representa por el rombo 'bloquea' que también se convertirá en una nueva tabla.

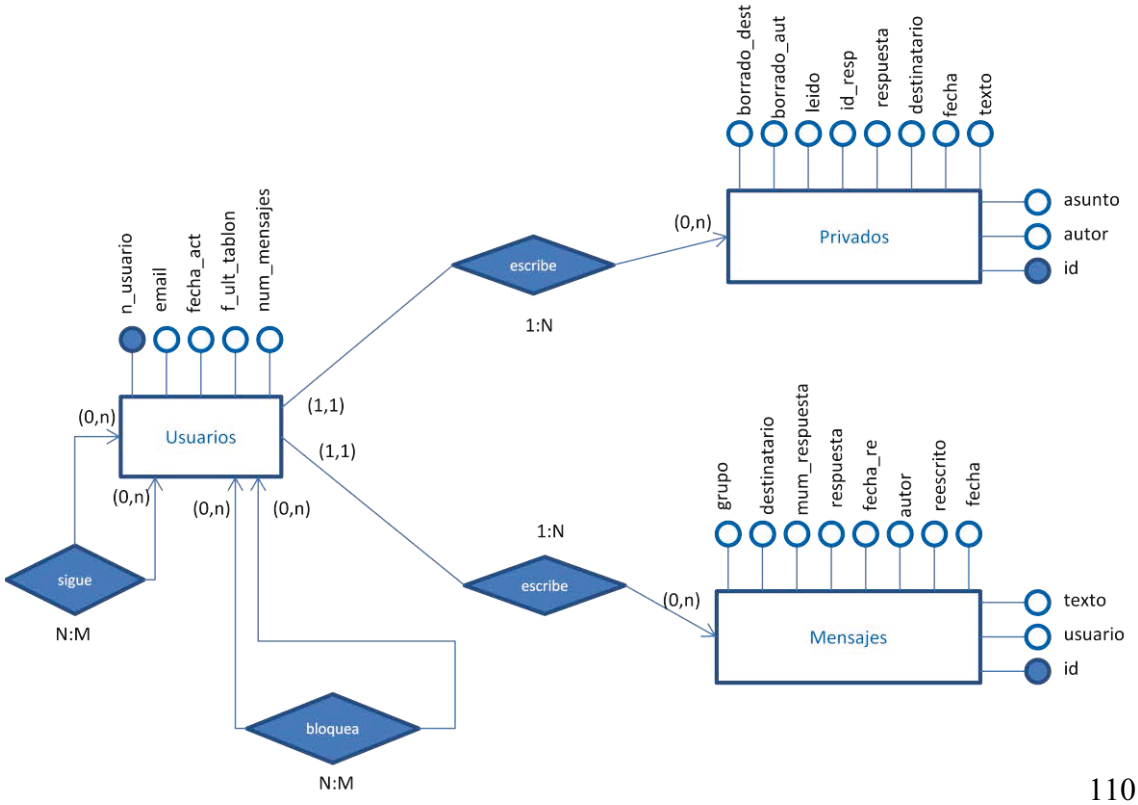


Ilustración 42. Diagrama Entidad-Relación

Las tablas se representan con cuadros blancos y las relaciones mediante rombos. De este modelo E-R se obtiene el grafo relacional que se muestra a continuación.

5.5.2 Grafo Relacional

En el grafo relacional se realiza la transformación del modelo E-R en tablas. En ellas se podrán observar las claves primarias (los atributos subrayados) y las claves externas o relaciones (indicadas con flechas).

Las relaciones 'N:M' del diagrama E-R se convierten en una nueva tabla cuya clave primaría será la formada por las dos claves primarias de las tablas que relacionan. Teniendo en cuenta esto el grafo queda de la siguiente forma.

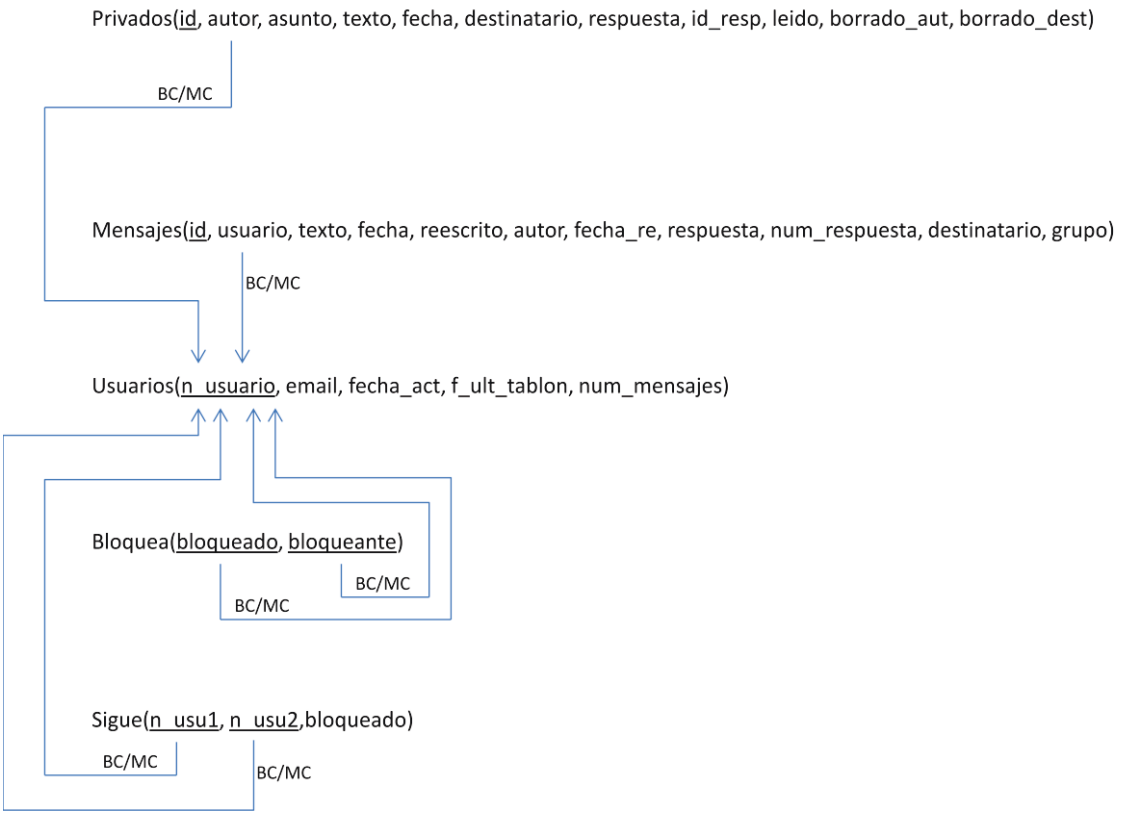


Ilustración 43. Grafo relacional

5.6 Diseño del directorio

5.6.1 Estructura de directorios.

En un principio el objetivo de este proyecto era poder utilizar el directorio de la universidad para poder validarse sobre él, pero debido a problemas de seguridad no se ha podido utilizar, por lo que para simular un funcionamiento similar, he optado por crear uno simulado sobre un servidor de directorios de ámbito local con una estructura de usuarios y grupos de asignaturas, el cual será utilizado para el funcionamiento del sistema. Más concretamente, los usuarios que quieran utilizar el sistema tendrán que validarse sobre él y cada usuario pertenecerá a los grupos indicados en dicho directorio.

Los directorios son especialmente eficientes en la lectura de datos, aunque también pueden hacerse modificaciones. En mi caso, el directorio siempre contendrá los mismos datos, la modificación sería para añadir nuevos usuarios matriculados o cambiar las asignaturas matriculadas una vez se cambiara el curso académico, pero eso no forma parte del objetivo del proyecto.

Un directorio es un conjunto de datos estructurados siguiendo una jerarquía en forma de árbol. Todos los directorios cuelgan de un nodo principal. Existe una página web con abundante información sobre directorios LDAP en la que me he apoyado para aprender y diseñar este directorio. [53] A continuación expongo y explicó la estructura del directorio que he creado.

Comenzaré explicando algunos conceptos para entender mejor el diagrama que expondré en el siguiente punto.

El directorio tiene dos tipos de nodos. Los nodos internos sirven para hacer divisiones y estructuran el árbol para clasificar los datos (organización o unidad de organización) y los nodos hoja son los datos.

Cada nodo consta de los siguientes elementos. Un nombre distintivo (DN) que se compone de su identificador propio junto con los DN de sus padres hasta llegar a la raíz. Uno o más objetos de clase que sirven para poder añadir los atributos que se desean. Cada objeto de clase permite añadir una serie de atributos concretos. [54] Por último los propios atributos en forma de tupla nombre-valor. Algunos atributos, según el objeto de clase son obligatorios y otros no. En el diagrama los atributos obligatorios aparecen en negrita.

A continuación se puede ver el diagrama que representa el directorio de la aplicación.

5.6.2 Diagrama del directorio

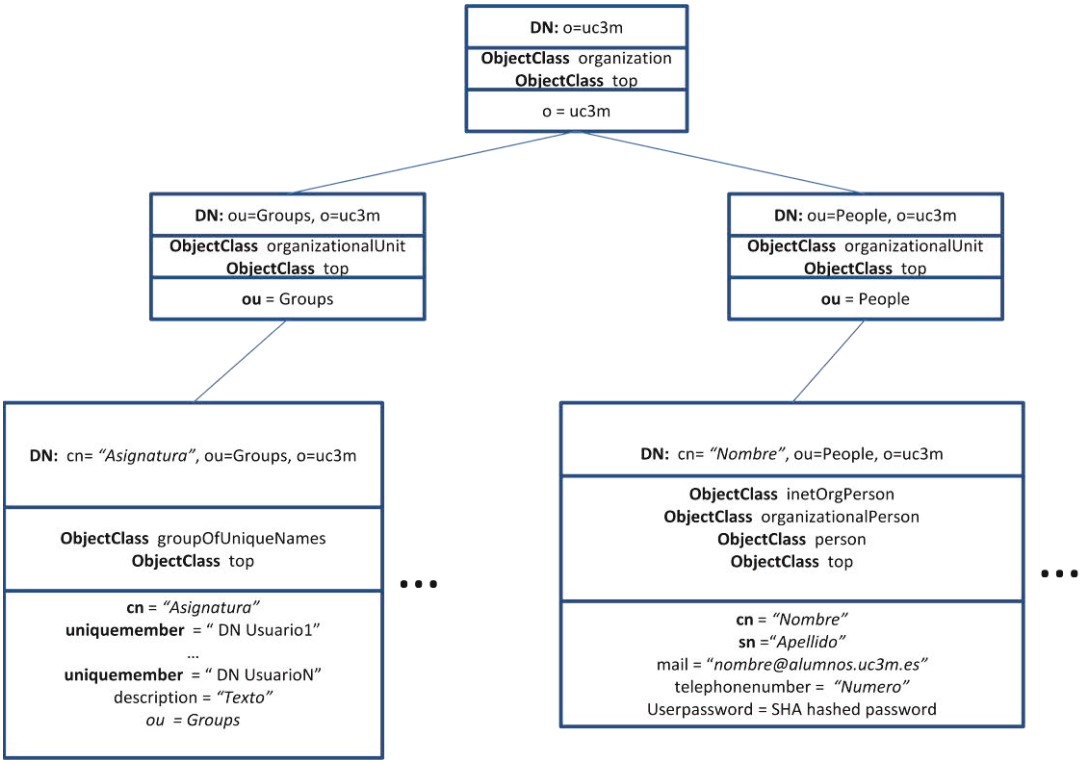


Ilustración 44. Diagrama del Directorio.

El nodo raíz se identifica como o=uc3m. Este nodo usa la clase 'organization' que se suele utilizar como raíz del directorio.

En el directorio se tienen que almacenar los usuarios y los grupos existentes, además es necesario almacenar la información referente sobre a qué grupos pertenece cada usuario. Para ello se han creado dos nuevos nodos que cuelgan del nodo uc3m. El nodo 'Groups' almacena los grupos y el nodo 'People' los usuarios. Ambos son de la clase organizationalUnit que suele usarse para crear nuevas divisiones para estructurar el directorio. De estos dos nodos cuelgan las entradas de datos. Las entradas que se muestran en el diagrama son generalizaciones, los nombres en cursiva se cambiarían por los valores concretos de cada atributo.

Las entradas de usuarios son de la clase 'person' que permite introducir atributos relacionados con los datos personales de una persona, el resto de clases son hijas de 'person' y permiten añadir otros atributos adicionales. No todos los atributos son importantes para el funcionamiento de la aplicación. Los atributos cn y Userpassword son las credenciales que permitirán hacer la autenticación y el atributo mail servirá para hacer búsquedas de usuarios por correo electrónico. El resto no son necesarios para el funcionamiento pero dan una idea mejor de la simulación del directorio.

Las entradas de grupos son de la clase 'groupOfUniqueNames' que permite añadir como atributo los identificadores de los usuarios que pertenecen a cada grupo (uniquemember). El valor de estos atributos será el DN de uno de los nodos de la rama People. De esta forma quedan relacionados los usuarios con los grupos.

Con esto queda claramente explicado cual es la estructura del directorio, qué almacena y como se relaciona con el sistema.

5.7 Conclusiones

Una vez que se tiene el diseño del sistema se puede empezar a desarrollar. En esta memoria no muestro detalles de código ya que sería imposible abarcar todo. Con el diseño que en este capítulo se muestra queda bastante claro en qué consiste el sistema.

A continuación voy a exponer las pruebas que se han realizado una vez terminada la fase de desarrollo para comprobar que todo funciona como se planteó en un principio.

Capítulo 6

Pruebas

6.1 Introducción

En este capítulo se van a diseñar una serie de pruebas sobre el sistema, que permitirán comprobar que todos los requisitos se cumplen a la finalización del desarrollo.

Se comienza probando la funcionalidad, para asegurarse de que el sistema permite hacer todo lo que se pensó en un principio. Finalmente se comprobará el resto de requisitos. Estas pruebas además de comprobar los requisitos, permitirá obtener posibles errores que no se hubieran detectado anteriormente.

Se van a exponer varias tablas con las pruebas y los resultados obtenidos para finalmente exponer una tabla con los requisitos conseguidos.

6.2 Diseño de pruebas

Se va a comenzar con las pruebas sobre la funcionalidad del sistema. Para el diseño de las pruebas se va a utilizar el documento de casos de uso, diseñando pruebas para cada uno, intentando cubrir todas las posibilidades que se pueden dar en cada uno de ellos.

A continuación se expone la estructura de las tablas con las definiciones de cada campo, en las que se definirán las pruebas.

ID caso de uso:	*Identificador del caso de uso.*	**Nombre caso de uso:**	*Nombre del caso de uso*	
Preparación:	*Lo que hay que realizar en el sistema para realizar la prueba.*			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	*Precondiciones del sistema para realizar la prueba individual correspondiente.*	*Acción del usuario sobre el sistema. Dependiendo de lo que realice se producirá una salida.*	*Lo que se prevé que el sistema hará en respuesta a la acción del usuario.*	*Si o no dependiendo de si se cumple lo esperado.*
2	*Precondiciones del sistema para realizar la prueba individual correspondiente.*	*Acción del usuario sobre el sistema. Dependiendo de lo que realice se producirá una salida.*	*Lo que se prevé que el sistema hará en respuesta a la acción del usuario.*	*Si o no dependiendo de si se cumple lo esperado.*

Tabla 2. Diseño tabla de pruebas.

Cada tabla recoge las pruebas realizadas sobre cada caso de uso. En la parte superior aparecen el identificador y el nombre del caso de uso que se corresponde con un identificador del un caso de uso del documento de casos de uso (Apéndice B). En el campo preparación se expone que hay que realizar en el sistema para poder llevar a cabo las pruebas correspondientes. En la parte inferior se definen las pruebas con su identificador estado previo del sistema, las acciones del usuario y la salida esperada. Para terminar se indica si la salida fue la esperada o no.

En el siguiente apartado se exponen las tablas con las pruebas diseñadas.

6.3 Pruebas funcionales.

ID caso de uso:	CU-01	**Nombre caso de uso:**	Autenticación	
Preparación:	Existe un usuario en el directorio con sus credenciales.			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Pantalla de acceso	El usuario introduce su nombre y contraseña correcta	Página principal	SI
2	Pantalla de acceso	El usuario introduce un nombre o contraseña incorrecta	Mensaje de aviso Introducir de nuevo	SI
3	Pantalla de acceso	El usuario introduce nombre o contraseña en blanco	Mensaje de aviso Introducir de nuevo	SI

Tabla 3. Prueba CU-01.

ID caso de uso:	CU-02	**Nombre caso de uso:**	Salir del sistema	
Preparación:	Existe un botón para salir del sistema.			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Página principal	El usuario pincha sobre el botón de salir	Cierra sesión Página de acceso	SI
2	Página principal	Se agota el tiempo de sesión	Cierra sesión Página de acceso	SI

Tabla 4. Prueba CU-02.

ID caso de uso:	CU-03	**Nombre caso de uso:**	Leer mensajes	
Preparación:	El usuario ha iniciado sesión.			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Página principal, el usuario no tiene mensajes	Ninguna	Tablón vacío	SI
2	Página principal, el usuario tiene 10 o menos mensajes.	Ninguna	Tablón muestra los mensajes sin opciones de cambio de página.	SI
3	Página principal, el usuario tiene más de 10 mensajes.	Ninguna	Tablón muestra los 10 mensajes más recientes y añade opciones de cambio de página.	SI

Tabla 5. Prueba CU-03.

ID caso de uso:	CU-04	Nombre caso de uso:	Filtrar mensajes	
Preparación:	El usuario ha iniciado sesión.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal, el usuario no tiene mensajes	Cualquier acción sobre el filtro	Tablón vacío	SI
2	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige un seguido concreto que ha escrito mensajes.	Tablón muestra los mensajes de ese usuario.	SI
3	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige un seguido concreto que no ha escrito mensajes.	Tablón vacío.	SI
4	Página principal, el usuario pertenece al menos a un grupo.	El usuario elige un grupo concreto en el que hay usuarios que han escrito mensajes	Tablón muestra los mensajes de ese grupo	SI
5	Página principal, el usuario pertenece al menos a un grupo.	El usuario elige un grupo concreto en el que los usuarios no han escrito mensajes	Tablón vacío	SI
6	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige una fecha concreta en el que uno de sus seguidos escribió algún mensaje	Tablón muestra todos los mensajes de seguidos que se escribieron en esa fecha.	SI
7	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige una fecha concreta en el que ninguno de sus seguidos escribió mensajes	Tablón vacío	SI
8	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige dos fechas para crear un intervalo de tiempo y alguno de sus seguidos ha escrito un mensaje entre esas dos fechas.	Tablón muestra todos los mensajes de seguidos que se escribieron entre esas fechas.	SI
9	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige dos fechas para crear un intervalo de tiempo y ninguno de sus seguidos ha escrito mensajes entre esas dos fechas.	Tablón vacío	SI
10	Página principal, el usuario pertenece al menos a un grupo.	El usuario elige un grupo concreto en el que hay usuarios que han escrito mensajes ,y elige un usuario concreto de ese grupo que ha escrito al menos un mensaje	Tablón muestra los mensajes de ese usuario concreto escritos para el grupo seleccionado.	SI
11	Página principal, el usuario pertenece al menos a un grupo.	El usuario elige un grupo concreto en el que hay usuarios que han escrito mensajes ,y elige un usuario concreto de ese grupo que no ha escrito mensajes	Tablón vacío	SI

12	Página principal, el usuario pertenece al menos a un grupo.	El usuario elige un grupo concreto en el que hay usuarios que han escrito mensajes , elige un usuario concreto de ese grupo que ha escrito al menos un mensaje, elige una fecha en la que ese usuario ha escrito al menos un mensaje	Tablón muestra los mensajes de ese usuario concreto, escritos para el grupo seleccionado, en la fecha seleccionada.	SI
13	Página principal, el usuario pertenece al menos a un grupo.	El usuario elige un grupo concreto en el que hay usuarios que han escrito mensajes , elige un usuario concreto de ese grupo que ha escrito al menos un mensaje, elige una fecha en la que ese usuario no ha escrito mensajes	Tablón vacío	SI
14	Página principal, el usuario pertenece al menos a un grupo.	El usuario elige un grupo concreto en el que hay usuarios que han escrito mensajes , elige un usuario concreto de ese grupo que ha escrito al menos un mensaje, elige un intervalo de tiempo en que ese usuario ha escrito al menos un mensaje	Tablón muestra los mensajes de ese usuario concreto, escritos para el grupo seleccionado, en el intervalo seleccionado.	SI
15	Página principal, el usuario pertenece al menos a un grupo.	El usuario elige un grupo concreto en el que hay usuarios que han escrito mensajes , elige un usuario concreto de ese grupo que ha escrito al menos un mensaje, elige un intervalo de tiempo en que ese usuario no ha escrito mensajes.	Tablón vacío	SI
16	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige un seguido concreto y una fecha concreta en el que dicho seguido escribió algún mensaje.	Tablón muestra los mensajes de ese seguido escritos en esa fecha.	SI
17	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige un seguido concreto y una fecha concreta en el que dicho seguido no escribió mensajes.	Tablón vacío	SI
18	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige un seguido concreto y un intervalo en el que dicho seguido escribió algún mensaje.	Tablón muestra los mensajes de ese seguido escritos en ese intervalo.	SI
19	Página principal, el usuario tiene seguidos y estos han escrito algún mensaje.	El usuario elige un seguido concreto y un intervalo en el que dicho seguido no escribió mensajes.	Tablón vacío	SI

Tabla 6. Prueba CU-04.

ID caso de uso:	CU-05	Nombre caso de uso:	Modificar página tablón	
Preparación:	Tiene que haber más de 10 mensajes en el tablón para que se hayan generado los botones de cambiar de página.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal. Hay más de 10 mensajes en el tablón. El usuario está viendo una página pero hay mensajes más antiguos.	El usuario pincha en siguiente	Tablón muestra los 10 mensajes siguientes más antiguos.	SI
2	Página principal. Hay más de 10 mensajes en el tablón. El usuario está en una página de mensajes antiguos.	El usuario pincha en anterior.	Tablón muestra los 10 mensajes anteriores más recientes.	SI
3	Página principal, el usuario tiene más de 10 mensajes.	El usuario selecciona un número de página concreto.	Tablón muestra los 10 mensajes de la página seleccionada.	SI

Tabla 7. Prueba CU-05.

ID caso de uso:	CU-06	Nombre caso de uso:	Buscar usuario	
Preparación:	El usuario estará autenticado en el sistema.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal.	El usuario pincha en el botón de buscar usuarios, selecciona la opción nombre y escribe un nombre a buscar. No hay ningún usuario con ese nombre.	La tabla de resultados estará vacía.	SI
2	Página principal.	El usuario pincha en el botón de buscar usuarios, selecciona la opción nombre y escribe un nombre a buscar. Existen usuarios con ese nombre.	La tabla de resultados mostrará los usuarios con ese nombre con sus datos y un botón de 'seguir usuario'.	SI
3	Página principal.	El usuario pincha en el botón de buscar usuarios, selecciona la opción email y escribe un email a buscar. No hay ningún usuario con ese email.	La tabla de resultados estará vacía.	SI
4	Página principal.	El usuario pincha en el botón de buscar usuarios, selecciona la opción email y escribe un email a buscar. Existen usuarios con ese nombre.	La tabla de resultados mostrará los usuarios con ese email con sus datos y un botón de 'seguir usuario'.	SI

Tabla 8. Prueba CU-06.

ID caso de uso:	CU-07	Nombre caso de uso:	Seguir usuario	
Preparación:	El usuario habrá buscado un usuario y la tabla de resultados esta rellena con al menos un usuario.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal. Tabla de usuarios mostrándose.	El usuario pincha sobre el botón de seguir.	El usuario se añade a la tabla de seguidos, se actualiza el tablón de mensajes y se muestra un mensaje de éxito.	SI
2	Página principal. Tabla de usuarios mostrándose.	El usuario pincha sobre el botón de seguir. Pero el usuario ya está siendo seguido.	Se muestra un mensaje de error.	SI

Tabla 9. Prueba CU-07.

ID caso de uso:	CU-08	Nombre caso de uso:	No seguir usuario	
Preparación:	El usuario estará autenticado en el sistema y estará siguiendo a un usuario por lo menos.			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Página principal.	El usuario pincha sobre el botón de no seguir. En la tabla de usuarios seguidos.	El usuario se elimina de la tabla de seguidos y se actualiza el tablón de mensajes.	SI

Tabla 10. Prueba CU-08.

ID caso de uso:	CU-09	Nombre caso de uso:	Bloquear usuario	
Preparación:	El usuario estará autenticado en el sistema y estará siendo seguido por un usuario al menos.			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Página principal.	El usuario pincha sobre el botón de bloquear en la tabla de usuarios seguidores.	El usuario se añade como bloqueado y se cambia el estilo como se muestra ese usuario cambiando el botón a desbloquear.	SI

Tabla 11. Prueba CU-09.

ID caso de uso:	CU-10	Nombre caso de uso:	Desbloquear usuario	
Preparación:	El usuario estará autenticado en el sistema y tendrá a algún usuario bloqueado.			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Página principal.	El usuario pincha sobre el botón de desbloquear, en la tabla de usuarios seguidores.	El usuario se elimina como bloqueado y se pone el estilo normal, añadiendo el botón bloquear.	SI

Tabla 12. Prueba CU-10.

ID caso de uso:	CU-11	Nombre caso de uso:	Ver seguidos	
Preparación:	El usuario estará autenticado en el sistema.			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Página principal.	El usuario pincha sobre el botón de ver todos los seguidos.	Se muestra la tabla con todos los usuarios a los que se está siguiendo.	SI
2	Página principal.	El usuario pincha sobre el botón de ver todos los seguidos, pero no se está siguiendo a nadie.	La tabla de seguidos estará vacía.	SI
3	Página principal.	Ninguna	Se puede ver en todo momento los cinco seguidos con más mensajes escritos.	SI

Tabla 13. Prueba CU-11.

ID caso de uso:	CU-12	Nombre caso de uso:	Ver seguidores	
Preparación:	El usuario estará autenticado en el sistema.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal.	El usuario pincha sobre el botón de ver todos los seguidores.	Se muestra la tabla con todos los usuarios que nos están siguiendo.	SI
2	Página principal.	El usuario pincha sobre el botón de ver todos los seguidos, pero ningún usuario nos sigue.	La tabla de seguidores estará vacía.	SI
3	Página principal.	Ninguna	Se puede ver en todo momento los cinco seguidores con más mensajes escritos.	SI

Tabla 14. Prueba CU-12.

ID caso de uso:	CU-13	Nombre caso de uso:	Escribir mensaje	
Preparación:	El usuario estará autenticado en el sistema.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal.	El usuario escribe un nuevo mensaje indicando en que grupo quiere escribir el mensaje y pincha en el botón escribir.	Se actualiza el tablón de mensajes y se muestra un mensaje de éxito. Se podrá ver el nuevo mensaje en el tablón del grupo seleccionado.	SI
2	Página principal.	El usuario escribe un nuevo mensaje indicando en que grupo quiere escribir el mensaje y pincha en el botón escribir. El mensaje tiene más de 140 caracteres.	Se muestra un mensaje de error.	SI

Tabla 15. Prueba CU-13.

ID caso de uso:	CU-14	Nombre caso de uso:	Borrar mensaje	
Preparación:	El usuario estará autenticado en el sistema y habrá escrito un mensaje por lo menos.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal.	El usuario pincha sobre el botón borrar mensaje que estará en el propio mensaje que se quiere borrar. El usuario tendrá que confirmar que quiere borrar el mensaje.	Si el usuario confirma el borrado, el mensaje se elimina y se actualiza el tablón de mensajes.	SI

Tabla 16. Prueba CU-14.

ID caso de uso:	CU-15	Nombre caso de uso:	Reescribir mensaje	
Preparación:	El usuario estará autenticado en el sistema y habrá un mensaje de otro usuario en el tablón de mensajes.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal.	El usuario pincha sobre el botón reescribir mensaje que estará sobre el propio mensaje.	Introduce el nuevo mensaje con información del autor original. Se actualiza el tablón de mensajes.	SI

Tabla 17. Prueba CU-15.

ID caso de uso:	CU-16	Nombre caso de uso:	Responder mensaje	
Preparación:	El usuario estará autenticado en el sistema y habrá un mensaje de otro usuario en el tablón de mensajes.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal.	El usuario pincha sobre el botón responder mensaje que estará sobre el propio mensaje, y rellena el formulario emergente para escribir su respuesta.	Se introduce la respuesta como un nuevo mensaje y se actualiza el tablón de mensajes.	SI
2	Página principal.	El usuario pincha sobre el botón responder mensaje que estará sobre el propio mensaje, y rellena el formulario emergente para escribir su respuesta. La respuesta tiene más de 140 caracteres.	Se muestra un mensaje de error.	SI

Tabla 18. Prueba CU-16.

ID caso de uso:	CU-17	Nombre caso de uso:	Enviar privado	
Preparación:	El usuario estará autenticado en el sistema. El usuario está siguiendo a algún usuario.			
ID - Prueba	Estado	Entrada acción del usuario	Salida esperada	Cumplimiento
1	Página principal.	El usuario pincha sobre el botón de enviar privado en el tablón de usuarios o en un mensaje del usuario al que se quiere enviar el privado. Se rellena el formulario emergente con los datos del privado.	Se envía el privado. El privado aparece en la tabla de privados enviados.	SI
2	Página principal.	El usuario pincha sobre el botón de enviar privado en el tablón de usuarios o en un mensaje del usuario al que se quiere enviar el privado. Se rellena el formulario emergente con los datos del privado. El mensaje supera los 300 caracteres.	Se muestra un mensaje de error.	SI

Tabla 19. Prueba CU-17.

ID caso de uso:	CU-18	Nombre caso de uso:	Leer privado	
Preparación:	El usuario estará autenticado en el sistema. El usuario habrá abierto el tablón de privados pinchando sobre ver privados. Habrá un privado.			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Página principal. Abierto el tablón de privados. Habrá un privado.	El usuario pincha sobre la pestaña de privados que desea ver, 'recibidos', 'leídos' o 'enviados' y pincha sobre el privado del que quiere ver el contenido.	Se muestra el contenido del mensaje.	SI
2	Página principal. Abierto el tablón de privados. Habrá un privado.	El usuario pincha sobre la pestaña de privados que desea ver, 'recibidos', 'leídos' o 'enviados' y pincha sobre el privado del que quiere ver el contenido. El privado leído es parte de una conversación.	Se muestra el contenido del mensaje, junto con todos los mensajes de la conversación. El mensaje leído aparece remarcado con otro estilo.	SI

Tabla 20. Prueba CU-18.

ID caso de uso:	CU-19	Nombre caso de uso:	Borrar privado	
Preparación:	El usuario estará autenticado en el sistema. El usuario habrá abierto el tablón de privados pinchando sobre ver privados. Habrá un privado			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Página principal. Abierto el tablón de privados. Habrá un privado	El usuario pincha sobre el botón borrar privado que estará en el propio privado que se quiere borrar. El usuario tendrá que confirmar que quiere borrar el privado.	Si el usuario confirma el borrado, el privado se elimina y se actualiza el tablón de privados.	SI

Tabla 21. Prueba CU-19.

ID caso de uso:	CU-20	Nombre caso de uso:	Responder privado	
Preparación:	El usuario estará autenticado en el sistema y habrá un privado de otro usuario en el tablón de privados.			
ID - Prueba	**Estado**	**Entrada acción del usuario**	**Salida esperada**	**Cumplimiento**
1	Página principal. Abierto el tablón de privados. Habrá un privado	El usuario pincha sobre el botón responder privado que estará sobre el propio privado, y rellena el formulario emergente para escribir su respuesta.	Se introduce la respuesta como un nuevo mensaje privado y se actualiza el tablón de privados.	SI
2	Página principal. Abierto el tablón de privados. Habrá un privado	El usuario pincha sobre el botón responder privado que estará sobre el propio privado, y rellena el formulario emergente para escribir su respuesta. La respuesta tiene más de 300 caracteres.	Se muestra un mensaje de error.	SI

Tabla 22. Prueba CU-20.

6.3 Consecución de requisitos.

Una vez realizadas las pruebas que comprueban la funcionalidad de la aplicación, se van a revisar los requisitos no funcionales obtenidos en la fase de análisis para chequear si se han cumplido. Para revisar los requisitos no funcionales he hecho una revisión del documento de requisitos (apéndice A) y he comprobado uno a uno si el objetivo del requisito a quedado plasmado en el sistema. Los requisitos de usabilidad no han sido comprobados con un grupo de usuarios, simplemente he comprobado que el requisito se cumple. Esto quiere decir que aunque los requisitos se cumplen, al no probar con un grupo de usuarios no se sabe si las soluciones de usabilidad propuestas son efectivas o no.

A continuación expongo las tablas con los requisitos y si se cumplen.

6.3.1 Consecución requisitos funcionales

ID	Nombre	Conseguido
RF-001	Autenticarse	SI
RF-002	Ver mensajes	SI
RF-003	Ver resumen seguidos	SI
RF-004	Ver número seguidos	SI
RF-005	Ver resumen seguidores	SI
RF-006	Ver número seguidores	SI
RF-007	Ver todos seguidos	SI
RF-008	Ver todos seguidores	SI
RF-009	Ver usuarios grupo	SI
RF-010	Buscar persona	SI
RF-011	Añadir seguido	SI
RF-012	Eliminar seguido	SI
RF-013	Bloquear seguidor	SI
RF-014	Desbloquear seguidor	SI
RF-015	Escribir mensaje	SI
RF-016	Escribir mensaje grupo	SI
RF-017	Eliminar mensaje	SI
RF-018	Reescribir mensaje	SI
RF-019	Responder mensaje	SI
RF-020	Filtrar mensajes nombre	SI
RF-021	Filtrar mensajes fecha	SI
RF-022	Filtrar mensajes Intervalo	SI
RF-023	Filtrar mensajes grupo	SI
RF-024	Aumentar página tablón mensajes	SI

RF-025	Disminuir página tablón mensajes	SI
RF-026	Ir a página tablón mensajes	SI
RF-027	Enviar privado	SI
RF-028	Ver tablón privados	SI
RF-029	Leer privado	SI
RF-030	Borrar privado	SI
RF-031	Responder privados	SI
RF-032	Ver respuestas	SI
RF-033	Ver mensaje respuesta	SI
RF-034	Usuario activo	SI
RF-035	Salir del sistema	SI
RF-036	Actualización automática	SI
RF-037	Contenido personalizado	SI

Tabla 23. Consecución requisitos funcionales.

6.3.2 Consecución requisitos de sistema

ID	Nombre	Conseguido
RS-001	Directorio	SI
RS-002	Compatibilidad navegadores web	SI
RS-003	Cumplimiento de estándares	SI
RS-004	BBDD	SI
RS-005	Tecnología de desarrollo	SI
RS-006	Lenguaje servidor	SI
RS-007	Librería Interfaz	SI
RS-008	Librería Ajax	SI
RS-009	IDE	SI

Tabla 24. Consecución requisitos de sistema.

6.3.3 Consecución requisitos de seguridad

ID	Nombre	Conseguido
RSG-001	Sesión	SI
RSG-002	Contraseñas cifradas	SI

Tabla 25. Consecución requisitos de seguridad.

6.3.4 Consecución requisitos de usabilidad

ID	Nombre	Conseguido
RU-001	Mensajes de operación	SI
RU-002	Estilo mensajes	SI
RU-003	Ayuda escritura mensajes color	SI
RU-004	Ayuda escritura mensajes numero caracteres.	SI
RU-005	Estilo usuarios bloqueados.	SI
RU-006	Estilo URL's	SI
RU-007	Mensajes confirmación borrado.	SI
RU-008	Estilo botón privados	SI
RU-009	Estilo tablón respuestas	SI
RU-010	Instrucciones	SI
RU-011	Maquetación líquida	SI
RU-012	Ver número página tablón mensajes	SI
RU-013	Ventanas modales	SI
RU-014	Mensajes de error	SI
RU-015	Links externos	SI
RU-016	Lenguaje familiar	SI
RU-017	Prevención de errores	SI
RU-018	Hilo respuestas privados	SI

Tabla 26. Consecución requisitos de usabilidad .

6.3.5 Consecución requisitos de interfaz

ID	Nombre	Conseguido
RI-001	Formulario autenticación	SI
RI-002	Formulario nuevo mensaje	SI
RI-003	Formulario buscar persona	SI
RI-004	Botón eliminar seguido	SI
RI-005	Botón bloquear	SI
RI-006	Botón eliminar mensaje	SI
RI-007	Botón reescribir mensaje	SI
RI-008	Botón responder mensaje	SI
RI-009	Formulario de respuesta	SI
RI-010	Filtro Select usuario	SI
RI-011	Filtro Select grupo	SI
RI-012	Filtro Input fechas	SI
RI-013	Botones paginación	SI

RI-014	Select paginación	SI
RI-015	Tabla privados	SI
RI-016	Botón enviar privado	SI
RI-017	Formulario privado	SI
RI-018	Botón leer privado	SI
RI-019	Botón responder privado	SI
RI-020	Botón eliminar privado	SI
RI-021	Tablón privados	SI
RI-022	Botón ver privados	SI
RI-023	Botón ver respuestas	SI
RI-024	Tablón respuestas	SI
RI-025	Enlace mensaje respuesta	SI
RI-026	Ventana respuesta	SI
RI-027	Botón salir	SI
RI-028	Elemento usuario	SI
RI-029	Página única	SI
RI-030	Interfaz Ajax	SI
RI-031	Barra de navegación	SI
RI-032	Barra lateral	SI
RI-033	Botones de acción	SI
RI-034	Botón seguir usuario	SI
RI-035	Botón ver seguidos	SI
RI-036	Botón ver seguidores	SI
RI-037	Reloj de sesión	SI

Tabla 27. Consecución requisitos de interfaz.

Como se puede observar se han conseguido todos los requisitos planteados en la fase de análisis.

6.4 Conclusiones

Con las pruebas queda cerrado el ciclo de desarrollo de la aplicación. En el siguiente capítulo se va a exponer el presupuesto del proyecto, dividiendo los costes en diferentes apartados.

Capítulo 7

Presupuesto

7.1 Introducción

En este capítulo se va a exponer un resumen con la planificación del proyecto, dividiéndolo en fases y sub-fases. También se incluirá el diagrama de Gantt que mostrará las diferentes actividades en contexto con el tiempo. Para terminar se va a realizar un desglose de los costes parciales y finalmente la suma total del coste del proyecto.

El calendario laboral que he utilizado es el estándar donde los días de trabajo van de Lunes a Viernes y los días de descanso los fines de semana. Cada jornada laboral consta de 4 horas de trabajo por lo que de media un mes tiene 80 horas de trabajo.

7.2 Planificación del proyecto

El proyecto ha tenido una duración de 180 días laborales, divididos en 6 fases diferentes.

La primera fase del proyecto es la fase de estudio preliminar. Incluye el estudio general del problema a resolver. Tras la primera reunión con el tutor y sus indicaciones, se realiza un estudio general para identificar las tecnologías y herramientas que son necesarias para el desarrollo del sistema. Una vez se han elegido, se estudian para familiarizarse con su uso y poder realizar las diferentes tareas de desarrollo de forma correcta. En esta fase también se incluye la instalación y la configuración del entorno de desarrollo, es decir, instalación de servidores, librerías, etc. y su configuración. Esta fase de instalación y configuración se ha realizado en paralelo con la fase de estudio mencionada.

La segunda fase es la fase de análisis donde se obtienen los requisitos y los casos de uso que nos dan una visión general de lo que realiza la aplicación dibujando una línea guía para la realización del proyecto.

La tercera fase es la fase de diseño. Apoyándose en los documentos obtenidos en la fase de análisis, se realiza el diseño de la arquitectura del sistema, el diseño estático y dinámico y el diseño de la BBDD y del directorio.

La cuarta fase es la fase de desarrollo que se divide en otras sub-fases que dividen el problema y hacen más fácil el desarrollar el sistema completo. El beneficio de dividir esta fase en sub-fases es que se simplifica el desarrollo y el posterior mantenimiento. Primero se definen los objetivos de cada sub-fase, se planifica y finalmente se desarrolla. Cuando cada sub-fase esta conforme a los objetivos planteados se inicia la siguiente. Cada sub-fase es una funcionalidad del sistema.

La última fase es la fase de pruebas en las que se han diseñado una serie de pruebas para asegurarse de que se cumplen todos los requisitos obtenidos en la fase de análisis, buscando a su vez posibles errores no detectados anteriormente.

La fase de documentación dura todo el proyecto y se divide en dos sub-fases, la documentación y la memoria. La fase de documentación consiste en la recolección de la información necesaria tanto de tecnologías como del análisis y diseño así como toda la información que crea necesaria para ayudar a realizar la memoria. La memoria se realiza al finalizar el desarrollo y las pruebas, y consiste en la revisión , agrupación y compleción de los datos recogidos durante toda las fases anteriores.

A continuación se puede ver gráficamente lo aquí mencionado, en el diagrama de Gantt.

7.3 Diagrama de Gantt

	Nombre de tarea	Duración
1	**PROYECTO**	**180 días?**
2	**FASE DE ESTUDIO PRELIMINAR**	**26 días**
3	Estudio del problema a resolver	3 días
4	Estudio librería DWR	3 días
5	Estudio LDAP y LDIF	6 días
6	Busqueda y estudio del servidor directorios	5 días
7	Estudio JNDI	9 días
8	Instalación y configuración del entorno de desarrollo	26 días
9	**FASE DE ANALISIS**	**15 días**
10	Obtención de requisitos	7 días
11	Obtención de casos de uso	8 días
12	**FASE DE DISEÑO**	**27 días**
13	Diseño de la arquitectura del sistema	6 días
14	Diseño estático	5 días
15	Diseño dinámico	5 días
16	Diseño de la BBDD	2 días
17	Diseño del directorio	9 días
18	**FASE DE DESARROLLO**	**74 días**
19	Creación de las tablas de la BBDD	2 días
20	Creación del directorio	3 días
21	Implementación de la interfaz de usuario	8 días
22	Implementación de la autenticación de usuarios	9 días
23	Implementación de operaciones de gestión de usuarios (seguir, no seguir, bloquear, desbloquear, buscar usuario)	9 días
24	Implementación de operaciones de gestion de mensajes(escribir, reescribir, responder, borrar mensajes)	11 días
25	Implementación del filtro de mensajes	12 días
26	Implementación de operaciones de gestión del tablón(suiguiente, anterior, ir a hoja)	4 días
27	Implementación de operaciones de gestión de privados(escribir, responder, leer, borrar privado)	9 días
28	Implementacion de operaciones de gestión de grupos	3 días
29	Implementación de la actualización automatica de la interfaz	4 días
30	**FASE DE PRUEBAS**	**7 días?**
31	Diseño de pruebas	3 días
32	Ejecución de pruebas	1 día?
33	Solución de errores	3 días
34	**FASE DE DOCUMENTACIÓN**	**180 días**
35	Documenación	149 días
36	Memoria	31 días

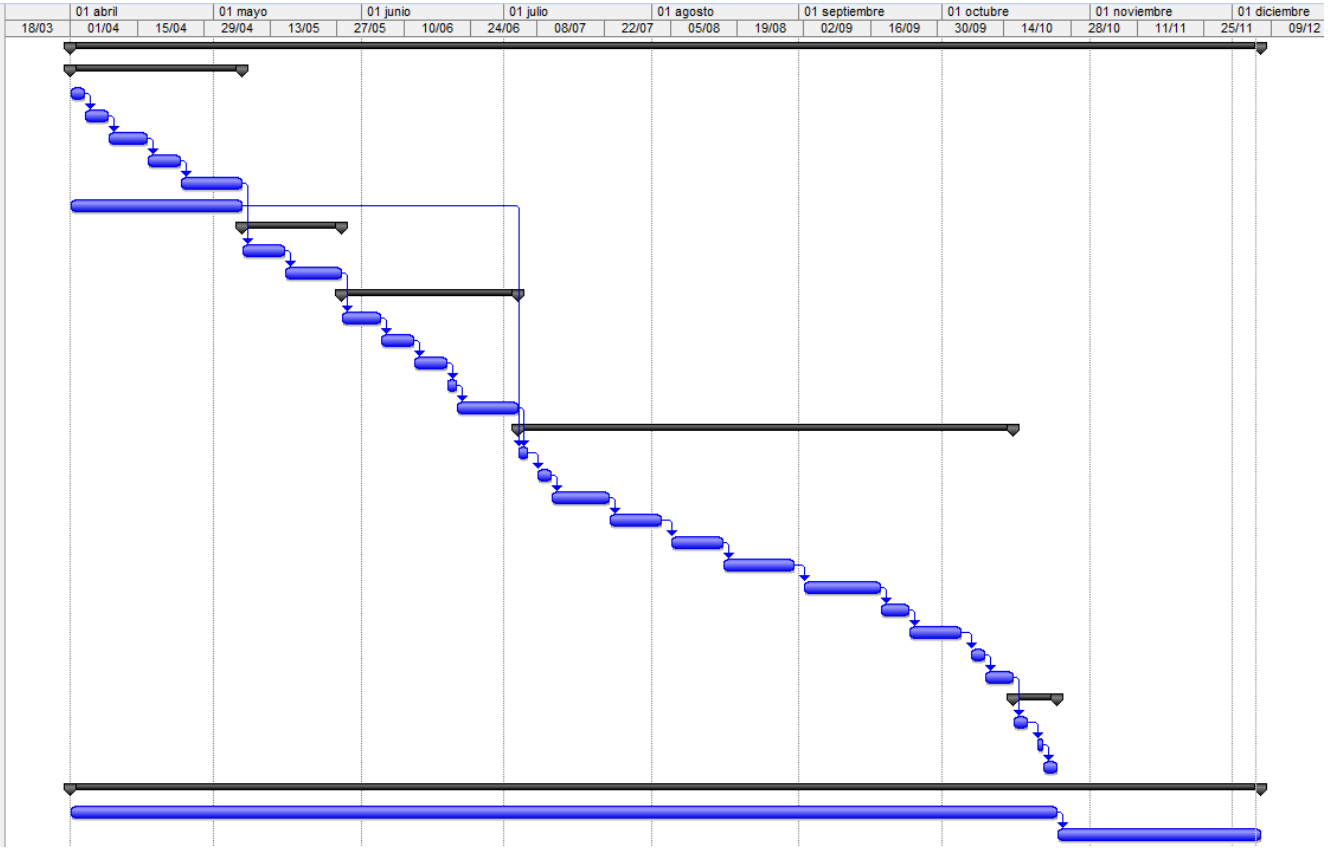


Ilustración 45. Diagrama de Gantt.

7.4 Desglose de costes

En este apartado se van a exponer de forma detallada los gatos del proyecto, desglosándolos en los diferentes tipos, coste de personal, costes de material y de software y licencias. Finalmente se expondrá el resumen general con el coste total del proyecto.

Se comenzará haciendo un resumen con las horas de cada fase del proyecto.

7.4.1 Horas dedicadas

En la siguiente tabla se pueden observar las horas dedicadas a cada fase del proyecto.

HORAS DEDICADAS	Días	Hora/día	Horas
FASE DE ESTUDIO	26	4	104
FASE DE ANALISIS	15	4	60
FASE DE DISEÑO	27	4	108
FASE DE DESARROLLO	74	4	296
FASE DE PRUEBAS	7	4	28
FASE DE DOCUMENTACIÓN	180	1	180
		TOTAL	776

Tabla 28. Tabla de horas dedicadas a cada fase.

El proyecto tiene una duración total de 180 días y 776 horas.

7.4.2 Coste de personal

Las horas de la tabla anterior se reparten entre los diferentes recursos de personal. Para realizar las diferentes fases o actividades del proyecto se han incluido 4 recursos de personal, entre ellos, un diseñador web, un analista, un ingeniero programador y un responsable de documentación. Se han considerado todos los recursos como "junior" al tener menos de 2 años de experiencia.

En la siguiente tabla se expone el resumen de horas por recurso, el precio hora estimado, la dedicación de hombres al mes y el coste total. El analista es el encargado de la fase de análisis, el diseñador web es el encargado de la fase de diseño, el ingeniero se encargara de desarrollar la aplicación además del estudio de las tecnologías y herramientas necesarias para ello y el responsable de documentación se encarga de todo lo relacionado con la documentación.

COSTE PERSONAL	Horas	Precio hora	Dedicación(hombre/mes)	Coste total(€)
Analista(junior)	60	18,75	0,75	1125
Diseñador Web(junior)	108	13,39	1,35	1446,12
Ingeniero Programador(junior)	428	15,17	5,35	6492,76
Resp. de documentación.	180	8,92	2,25	1605,6
				10669,48

Tabla 29. Tabla de costes de personal.

Para calcular el precio hora he investigado en internet los salarios de empleados en puestos similares en diferentes comunidades autónomas y he hecho una estimación aproximada. Cada hombre en un mes realiza 80h de trabajo, por lo que es fácil obtener la dedicación. Tras hacer los cálculos y hacer la suma general el coste de personal asciende a 10669,48€ sin IVA.

7.4.3 Coste de material

En este apartado se exponen los costes de material. Para obtener la amortización se utiliza la siguiente fórmula, obtenida de la plantilla de presupuesto proporcionada por la UC3M.

$$\frac{A}{B} xCxD$$

A: número de meses desde la fecha de facturación en que el equipo es utilizado

B: periodo de depreciación = 60 meses.

C: coste del equipo.

D: porcentaje del uso que se dedica al proyecto.

COSTE HARDWARE	UNID	Coste	Coste sin IVA	Dedicación	Coste imputable(€)
Ordenador HP Pavilion DS6600	1	825	676,5	8 meses	90,2
Ratón óptico	1	5	4,1	8 meses	0,546666667
Impresora HP Laserject CP1215	1	250	205	1,5 meses	5,125
Pendrive TDK 8Gb	1	40	32,8	8 meses	4,373333333
Disco duro externo WD 500GB	1	90	73,8	8 meses	9,84
					110,085

Tabla 30. Tabla de coste de Hardware.

Tras el cálculo de la amortización del material utilizado el coste asciende a 110€ sin IVA.

7.4.4 Coste de Software y licencias.

En este apartado se exponen los costes del software utilizado para la realización del proyecto. Se incluyen desde el software necesario para el desarrollo, como el IDE de desarrollo, los servidores, o las librerías, así como el software para realizar la documentación, como procesadores de textos, diseño de gráficos y esquemas o software para gestión de proyectos. Se ha intentado desarrollar el proyecto utilizando licencias de software gratuitas, y, en su mayoría, se ha conseguido exceptuando el paquete Microsoft Office y el Project.

SOFTWARE Y LICENCIAS	UNID	Coste unid(€)	Coste Total (€)
Microsoft Office Professional 2007	1	181,7	181,7
Microsoft Office Project 2007	1	82.95	82.95
Editor Notepad++	1	0	0
StarUML	1	0	0
Dia	1	0	0
JBoss Aplication Server	1	0	0
MySql Server	1	0	0
Apache Directory Server	1	0	0
DWR	1	0	0
JQuery	1	0	0
JQuery UI	1	0	0
Eclipse	1	0	0
			264,65

Tabla 31. Tabla de coste de Software.

El coste total de software asciende a 264,65€ sin IVA.

7.4.5 Coste de material consumible

En este apartado se exponen los coste de materiales que se consumen con el uso.

MATERIAL	UNID	Coste unid (€)	Coste Total (€)
Folios(paquete)	1	2,37	2,37
Lápices	4	0.79	3,16
Gomas	2	0,2	0,4
Carpeta	1	2,37	2,37
Libreta	1	3,95	3,95
Recambios toner impresora	3	45	135
			147,25

Tabla 32. Tabla de coste de consumibles.

El coste total de material consumible asciende a 150,4€ sin IVA.

7.4.6 Coste Total del proyecto.

En este apartado se suman las cifras de costes vistas hasta ahora y se añaden los costes indirectos para obtener el coste total del proyecto completo. Los costes indirectos se definen en un 20% del coste total. Este porcentaje lo he obtenido de la plantilla para el presupuesto proporcionada por la UC3M.

COSTES	EUROS
Coste personal	10669,48
Coste hardware	110,085
Coste software	264,65
Coste consumibles	147,25
Costes indirectos (20%)	2238,293
	13429,758

Tabla 33. Tabla de coste total del proyecto.

El precio obtenido es el coste total sin IVA.

En el siguiente apartado se va a exponer un resumen del coste total del proyecto.

7.5 Resumen

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Ignacio Vidal Hernando
2.- Departamento:
Informática
3.- Descripción del Proyecto:

- Titulo	**Sistema Microblogging.**
- Duración (meses)	8
Tasa de costes Indirectos:	20%

4.- Presupuesto total del Proyecto (valores en Euros):
13.429,76 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Ignacio Vidal Hernando		Analista Junior	0,75	1.500,00	1.125,00	
Ignacio Vidal Hernando		Diseñador Junior	1,35	1.071,20	1.446,12	
Ignacio Vidal Hernando		Ingeniero Junior	5,35	1.213,60	6.492,76	
Ignacio Vidal hernando		R. Documentación	2,25	713,60	1.605,60	
		Hombres mes 9,7		**Total**	10.669,48	

* Calculado utilizando 1 Hombre mes = 80 horas
[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenador HP Pavilion DS6600	676,50	100	8	60	90,20
Ratón óptico	4,10	100	8	60	0,55
Impresora HP Laserject CP1215	205,00	100	2	60	5,13
Pendrive TDK 8Gb	32,80	100	8	60	4,37
Disco duro externo WD 500GB	73,80	100	8	60	9,84
					0,00
				Total	110,09

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} \text{xCxD}$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Software y Licencias		264,65
Material fungible		147,25
	Total	411,90

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas,

6.- Resumen de costes

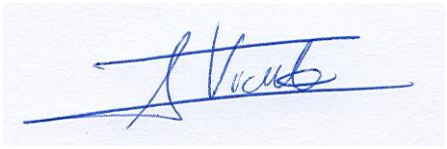
Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	10.669
Amortización	110
Subcontratación de tareas	0
Costes de funcionamiento	412
Costes Indirectos	2.238
Total	13.430

Ilustración 46. Resumen coste total del proyecto.

El presupuesto total de este proyecto asciende a la cantidad de TRECEMIL CUATROCIENTOS TREINTA EUROS.

Colmenarejo a 24 de Septiembre de 2013

El ingeniero proyectista



Fdo. Ignacio Vidal Hernando

Capítulo 8

Conclusiones y trabajo futuro

8.1 Introducción

En este capítulo se exponen las conclusiones obtenidas del trabajo realizado para el desarrollo del proyecto, teniendo en cuenta los objetivos planteados en el capítulo 1 de esta memoria. También se exponen posibles trabajos futuros que se podrían desarrollar y añadir al proyecto para mejorarlo y hacerlo más completo.

8.2 Conclusiones

El principal objetivo del proyecto era desarrollar una aplicación web de microblogging que permita la comunicación entre alumnos, y la comunicación con los diferentes profesores que usen la aplicación, y de esta forma disponer de una nueva herramienta docente diferente de las que se tienen hasta ahora, (foros, correo, etc.). Para ello, en un principio, se pretendía poder integrar la aplicación con el directorio LDAP de la universidad para que todos los usuarios contenidos en él pudieran usar la aplicación. Finalmente, por motivos de seguridad, no fue posible el uso del directorio real, por lo que se planteó crear un directorio simulado, instalado en un servidor de directorios en un entorno local.

No solo se pretendía crear la aplicación, sino que uno de los objetivos era crear una aplicación moderna, intuitiva y altamente interactiva, que permitiera usarla sin problemas desde un principio, sin casi tiempo de aprendizaje. Para ello se ha hecho un fuerte hincapié en el tema de la usabilidad.

Para desarrollar una aplicación web altamente interactiva es casi imprescindible usar la tecnología Ajax, y por ello en el planteamiento del proyecto se impuso como un objetivo a cumplir el desarrollar la aplicación con esta tecnología.

Previo a la realización del proyecto se hizo un profundo estudio sobre Ajax y las tecnologías que incluye. Toda la información obtenida de este estudio se reunió y documento en un trabajo dirigido que me sirvió para convalidar algunos créditos de libre elección, pero el objetivo principal de este estudio fue la preparación para el proyecto que realizaría en el futuro.

En este estudio comencé estudiando las diferentes tecnologías que conforman Ajax a más bajo nivel para ver y entender cómo funciona y tener una visión más global. La segunda parte del estudio era la elección de una librería que implementara Ajax y que facilitaría el desarrollo de la aplicación. Para la elección de esta librería estudié varias de las más populares, eligiendo finalmente aquella con la que me sentí más cómodo.

La librería que elegí para implementar las llamadas asíncronas Ajax fue DWR que es una librería que trabaja con Java por lo que decidí desarrollar la aplicación con este lenguaje. Durante la carrera no he visto Java en profundidad por lo que tuve que estudiar las nociones básicas de este lenguaje, pero en general fui aprendiendo mientras iba programando, estudiando la parte que me interesaba.

Al finalizar este proyecto puedo decir que se han cumplido los objetivos planteados.

El trabajar en el proyecto me ha proporcionado nuevos conocimientos sobre algunos aspectos:

- Ajax: antes de empezar el estudio no tenía ningún conocimiento sobre esta tecnología y ahora puedo decir que mi conocimiento en este tema es moderado.
- Java: conocimiento suficiente para programar cómodamente en Java. Java es tan extenso que es difícil conocer todo en profundidad pero utilizando la documentación [55] es fácil ir aprendiendo poco a poco.
- Instalación, configuración, y manejo de Eclipse: ya tenía algún conocimiento sobre Eclipse pero después de trabajar en el proyecto he afianzado bastantes conocimientos y he aprendido a usarlo de forma más eficiente.
- Configuración de un DataSource: he aprendido a configurar un DataSource para poder conectar a una base de datos.
- Nociones sobre directorios LDAP e instalación y configuración de un servidor LDAP: tampoco tenía conocimientos sobre LDAP antes de la realización de este proyecto. Tras terminarlo puedo decir que he adquirido los conocimientos básicos para el manejo de directorios.
- JDBC y JNDI: he adquirido conocimientos básicos sobre estas dos librerías para trabajar con bases de datos y directorios.
- Esquemas de directorios Ldif: he aprendido lo básico para crear esquemas de directorios utilizando este tipo de archivos y su lenguaje correspondiente.
- Ficheros properties: he adquirido conocimientos de cómo usar este tipo de ficheros y sus beneficios.

En el siguiente apartado habló sobre los posibles trabajos futuros que se podrían añadir al proyecto para hacerlo más completo.

8.2 Trabajo futuro

La aplicación que he desarrollado funciona en un entorno local. En un principio se pretendía que estuviera integrado con el directorio de la universidad, pero por motivos de seguridad no fue posible, por lo que adopte la solución de crear un directorio local y simular un posible esquema con alumnos y asignaturas para poder manejar la información. Dicho esto se podrían desarrollar otros trabajos para aumentar la funcionalidad y completar la aplicación.

- Integración de la aplicación en Aula global con la plataforma moodle y con el directorio LDAP de la universidad para poder usarlo con sus usuarios y asignaturas.
- Nuevas funcionalidades.
    - Introducción de diferentes roles de usuario, (alumnos, profesor , administrador) con funciones adicionales para según qué rol.
    - Nuevas formas de filtrar mensajes, por ejemplo buscar mensajes por palabra clave.
    - Añadir funcionalidades similares a las que usa Twitter como etiquetado de mensajes, referencias a otros usuarios o tendencias.
    - Añadir opciones de personalización, como una página de perfil donde añadir información personal, o una fotografía.

- Usabilidad
    - En el caso de añadir opciones de personalización se podrían personalizar los mensajes con una foto para diferenciarlos de forma más visible.
    - Añadir opciones de accesibilidad para personas con algún tipo de deficiencia.
- Otros trabajos
    - Creación de una aplicación que permitiera el acceso a través teléfonos móviles o tabletas.

Glosario

AJAX	*Asynchronous JavaScript And XML*	*JavaScript asíncrono y XML*
API	*Application Programming Interface*	*Interfaz de programación de aplicaciones*
ASP	*Active Server Pages*	
BBDD	*Base de datos*	
CSS	*Cascading Style Sheets*	*Hojas de estilo en cascada*
DAO	*Data Access Object*	
DHTML	*Dynamic HyperText Markup Language*	
DOM	*Document Object Model*	*Modelo de Objetos del Documento*
DWR	*Direct Web Remoting*	
GPL	*General Public License*	*Licencia Pública General*
HTML	*HyperText Markup Language*	*Lenguaje de marcas de hipertexto*
HTTP	*Hypertext Transfer Protocol*	*Protocolo de transferencia de hipertexto*
IDE	*Integrated development environment*	*Entorno de desarrollo integrado*
JDBC	*Java DataBase Connectivity*	
JDK	*Java Development Kit*	
JNDI	*Java Naming Directory Interface*	*Interfaz de Nombrado y Directorio Java*
JSON	*JavaScript Object Notation*	
JSP	*JavaServer Pages*	
LDAP	*Lightweight **D**irectory **A**ccess **P**rotocol*	*Protocolo Ligero de Acceso a Directorios*
LDIF	*LDAP Data Interchange Format*	
LPGL	*Lesser General Public License*	*Licencia Pública General Reducida*
PHP	*PHP Hypertext Pre-processor*	
SQL	*Structured query language*	*Lenguaje de consulta estructurado*
UML	*Unified Modeling Language*	*Lenguaje Unificado de Modelado*
URI	*Uniform Resource Identifier*	*Identificador uniforme de recursos*
XML	*eXtensible Markup Language*	*Lenguaje de marcas extensible*

Referencias

Se listan las referencias ordenadas numéricamente según su aparición en el texto.

[1] **Microblogging - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/Microblogging.

[2] **Web - Eclipse** [En línea]. - http://www.eclipse.org/.

[3] **Web - Java JDK** [En línea]. - http://www.oracle.com/technetwork/es/java/javasebusiness/downloads/index.html.

[4] **Web - DWR** [En línea]. - http://directwebremoting.org/dwr/index.html.

[5] **Web - JQuery** [En línea]. - http://jquery.com/.

[6] **Web - JQuery UI** [En línea]. - http://jqueryui.com/.

[7] **Web - JDBC Tutorial** [En línea]. - http://docs.oracle.com/javase/6/docs/technotes/guides/jdbc/.

[8] **Web - JNDI Tutorial** [En línea]. - http://docs.oracle.com/javase/tutorial/jndi/index.html.

[9] **Web - JBoss** [En línea]. - http://www.jboss.org/overview/.

[10] **Web - ApacheDS** [En línea]. - http://directory.apache.org/.

[11] **Web - MySql** [En línea]. - http://www.mysql.com/.

[12] **Web - StarUML** [En línea]. - http://staruml.sourceforge.net/en/.

[13] **Web - Dia** [En línea]. - http://dia-installer.de/index.html.es.

[14] **HTTP - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

[15] **Un nuevo enfoque de las Aplicaciones Web** [En línea] / aut. Garrett Jesse James. - http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/.

[16] **w3schools - html** [En línea]. - http://www.w3schools.com/html/.

[17] **w3schools - css** [En línea]. - http://www.w3schools.com/css/.

[18] **w3schools - javascript** [En línea]. - http://www.w3schools.com/js/.

[19] **w3schools - DOM** [En línea]. - http://www.w3schools.com/js/js_htmldom.asp.

[20] **w3schools - XML** [En línea]. - http://www.w3schools.com/xml/.

[21] **w3schools - XMLHTTPRequst** [En línea]. - http://www.w3schools.com/xml/xml_http.asp.

[22] **Ajax - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/AJAX.

[23] **Ventajas y desventaja Ajax** [En línea] / aut. Baldota Pritam. - http://www.pritambaldota.com/index.php/what-is-ajax/.

[24] **Videotutorial Ajax** [En línea] / aut. Núñez Jesús-Eduardo Conde // Videotutoriales.com. - http://www.youtube.com/watch?v=o2V5Ls7e9r4&list=PLF49E524856DAD8F7.

[25] **Librosweb Ajax** [En línea]. - http://librosweb.es/ajax/.

[26] **w3schools - Ajax** [En línea]. - http://www.w3schools.com/ajax/.

[27] **Professional Rich Internet Applications: AJAX and Beyond** [Libro] / aut. Moore Dana, Budd Raymond y Benson Edward. - [s.l.] : Wrox, March 2007. - ISBN: 978-0-470-08280-5.

[28] **Web - Prototype** [En línea]. - http://prototypejs.org/.

[29] **Web - Scriptaculous** [En línea]. - http://script.aculo.us/.

[30] **Web - Mootools** [En línea]. - http://mootools.net/.

[31] **Web - Openrico** [En línea]. - http://openrico.org/.

[32] **Web - GWT** [En línea]. - http://www.gwtproject.org/.

[33] **Servlet - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/Java_Servlet.

[34] **URI - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/Uniform_Resource_Identifier.

[35] **JavaBean - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/JavaBean.

[36] **Web - DWR Documentación** [En línea]. - http://directwebremoting.org/dwr/documentation/index.html.

[37] **DWR Java Ajax Applications** [Libro] / aut. Salkosuo Sami. - Birmingham : Packt Publishing Ltd, Octubre 2008. - ISBN 978-1-847192-93-6.

[38] **Web - JQuery Documentación** [En línea]. - http://api.jquery.com/.

[39] **w3schools - JQuery** [En línea]. - http://www.w3schools.com/jquery/.

[40] **Desarrolloweb - jQuery tutorial** [En línea]. - http://www.desarrolloweb.com/manuales/manual-jquery.html.

[41] **Learning jQuery** [Libro] / aut. Chaffer Jonathan y Swedberg Karl. - [s.l.] : Packt Publishing, 2011. - 3. - Disponible en linea: http://it-ebooks.info/book/1248/.

[42] **Web - JQuery UI Themeroller** [En línea]. - http://jqueryui.com/themeroller/.

[43] **Web - JQuery UI Documentación** [En línea]. - http://api.jqueryui.com/.

[44] **JQuery UI - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/JQuery_UI.

[45] **LDIF - Wikipedia** [En línea]. - http://en.wikipedia.org/wiki/LDAP_Data_Interchange_Format.

[46] **LDAP - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/LDAP.

[47] **SHA - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/Secure_Hash_Algorithm.

[48] **The Design of Sites: Patterns for Creating Winning Web Sites** [Libro] / aut. Duyne Douglas K. van, Landay James A. y Hong Jason I.. - [s.l.] : Prentice Hall, 2006. - Disponible en linea: http://my.safaribooksonline.com/book/web-design-and-development/0131345559. - ISBN 978-0-13-134555-3.

[49] **Cliente - Servidor - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/Cliente-servidor#Arquitecturas_multi-capas.

[50] **DAO - Wikipedia** [En línea]. - http://es.wikipedia.org/wiki/Data_Access_Object.

[51] **Web - Java Datasource** [En línea]. - http://docs.oracle.com/javase/tutorial/jdbc/basics/sqldatasources.html.

[52] **Properties - Wikipedia** [En línea]. - http://en.wikipedia.org/wiki/.properties.

[53] **Web - LDAP Tutorial** [En línea]. - http://www.zytrax.com/books/ldap/.

[54] **Web - Atributos y objetos LDAP** [En línea]. - http://www.zytrax.com/books/ldap/ape/.

[55] **Piensa en Java** [Libro] / aut. Eckel Bruce. - [s.l.] : Prentice Hall.

Apéndices

Apéndice A: Requisitos de software.

Introducción

Propósito

El objetivo de este documento es recoger los requisitos del sistema de Microblogging que se desarrollará, y las matrices de trazabilidad de estos con los casos de uso.

Personal involucrado

Nombre	Ignacio Vidal Hernando
Rol	Analista.
Categoría profesional	Junior.
Responsabilidades	Se encargará de la recogida y análisis de requisitos.

Acrónimos y definiciones

En este apartado se incluye la lista de acrónimos y definiciones que se utilizarán en el documento.

Acrónimos

- **RS-F:** requisito de software funcional.
- **RS-S:** requisito de software de sistema.
- **RS-SG:** requisito de software de seguridad.

- **RS-U:** requisito de software de usabilidad.
- **RS-I:** requisito de software de interfaz.

- **AJAX:** Asynchronous JavaScript And XML.
- **XML:** eXtensible Markup Language.
- **HTML:** HyperText Markup Language.
- **CSS:** Cascading Style Sheets.
- **DWR:** Direct Web Remoting.
- **W3C:** World Wide Web Consortium.

Definiciones

- **Datepicker**: es un widget de creación de interfaces de usuario proporcionado por la librería JQuery User Interface, que consiste en un campo de entrada que cuando recoge el foco se convierte en un pequeño calendario, pudiendo elegir una fecha. De esta forma se evitan problemas con el formato de fecha eliminando gran cantidad de errores y haciendo más fácil el uso de la aplicación para el usuario.
- **Select**: es un campo de entrada dentro de las interfaces de usuario que consiste en una caja que cuando recibe el foco se abre ofreciendo una serie de opciones al usuario.
- **Feedback**: el feedback en la web se refiere a la reentrada de información por parte del sistema para ayudar al usuario a que la navegación sea más sencilla ya que en todo momento estará informado de lo que está haciendo la aplicación.
- **Directorio**: es un conjunto de objetos que tienen una serie de atributos y que están organizados de forma jerárquica y lógica.

Resumen

En el resto del documento se incluirá una descripción general del producto a desarrollar. La parte principal del documento constará de la descripción de los requisitos, cada uno en una tabla con diferentes atributos que permitan verificar el producto final de forma clara y simple. Se dividirán en 5 tipos, requisitos de usuario, requisitos de sistema, requisitos de seguridad, requisitos de usabilidad y requisitos de interfaz. Además se incluirá una tabla de trazabilidad entre los requisitos y los casos de uso, de este modo se podrá obtener la información de relación entre los requisitos y cada caso de uso de forma rápida.

Descripción general

Perspectiva del producto

El producto que se va a desarrollar no forma parte de un sistema mayor, es un producto independiente.

Funcionalidad del producto

El producto consiste en una aplicación de interacción entre usuarios mediante mensajes asíncronos vía web. Los usuarios escribirán sus mensajes para que los demás puedan leerlos en el momento que se conecten a la aplicación, no será necesario crear una conexión directa entre dos usuarios para que puedan comunicarse, es una comunicación todos con todos pero de forma asíncrona.

Cada mensaje estará unido a un alumno o profesor como usuarios de la aplicación y autores de los mensajes. A su vez, estos mensajes podrán estar unidos a una asignatura con el objetivo de que todos los mensajes de esa asignatura se muestren de forma conjunta, pero no es siempre necesario que estén unidos a una asignatura, también se pueden escribir mensajes sin relación a ningún grupo, de esta forma también se puede usar la aplicación de forma más libre, siguiendo a los usuarios que se desee para ver sólo sus mensajes.

Los usuarios de la aplicación podrán buscar a otros usuarios y seguirlos para poder ver sus mensajes, a su vez estos podrán ser seguidos por otros de forma que puedan leer también sus propios mensajes.

Se podría decir que la aplicación tiene dos funcionalidades principales, interacción de entrada y de salida, es decir escribir sus propios mensajes y leer los mensajes de los demás usuarios.

Los usuarios podrán escribir mensajes de hasta 140 caracteres, pero a su vez podrán reescribir o responder a mensajes de otros usuarios. Los mensajes serán de dos tipos, públicos y privados. Los mensajes públicos podrán ser leídos por todos los usuarios que estén siguiendo al autor de los mensajes, mientras que los privados solo serán visibles para los usuarios a los que vayan dirigidos.

Como se ha comentado el producto tiene la posibilidad de agrupar usuarios en diferentes grupos de asignaturas que serán provistos de forma personalizada para cada

usuario del sistema dependiendo de qué asignaturas tenga matriculadas en cada curso, de esta forma se podrán seguir los mensajes relacionados con una asignatura de una forma mucho más concreta y sencilla. Así, por ejemplo, el profesor de la asignatura y los alumnos podrían usar la aplicación para crear debates, hacer comentarios o simplemente ayudarse a resolver dudas sobre algún aspecto de la asignatura.

Estas son las funcionalidades principales pero habrá otras complementarias que ayudarán a que la experiencia sea más completa y satisfactoria. Todas las funcionalidades se recogen en las tablas de requisitos que se pueden encontrar a continuación.

Requisitos de software

En este apartado se definirán todos los requisitos del producto a desarrollar separados en diferentes tipos, funcionales, sistema, seguridad, usabilidad e interfaz. Cada requisito tendrá una tabla con una serie de atributos que se exponen a continuación:

- Identificador: es el identificador del requisito, se compondrá de unos caracteres que indicarán el tipo y un número.
- Nombre: es la identificación en modo texto del requisito.
- Necesidad: Indica la importancia del requisito dentro del proyecto. Se seleccionará una de las tres casillas disponibles (esencial, deseable u opcional), siendo esencial si el requisito es de máxima necesidad, opcional si no es muy importante, y deseable si es de nivel medio.
- Prioridad: se dispondrá de tres niveles, alto, medio o bajo. El nivel alto indica que el requisito es imprescindible y no puede cambiar en el futuro. La prioridad media indica que es importante para el proyecto pero podrá sufrir cambios en el futuro. La prioridad baja indica que el requisito es menos importante y que puede llegar a ser modificado o incluso eliminado en el futuro.
- Patrón: si el requisito está basado en alguno de los patrones del libro '*The Design of Sites: Patterns for creating Winning Websites, Second Edition*' [1] se indicará en cuál. Se hará mediante una letra que identifica el tema del patrón, un número que indica el problema que se resuelve dentro de dicho tema y el nombre de este.
- Descripción: la descripción explica de forma textual en qué consiste el requisito, intentando ser lo suficientemente clara y completa para que se entienda perfectamente y no queden dudas al respecto.

Requisitos funcionales

Identificador: RF-001	
Nombre: Autenticarse	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: H2 Autenticarse (Ayudando a los usuarios a completar tareas)	
Descripción: El usuario podrá autenticarse en el sistema.	

Tabla 34. RF-001 Autenticarse.

[1] **Duyne, Douglas K. van, Landay, James A. and Hong, Jason I.** *The Design of Sites: Patterns for Creating Winning Web Sites.* Segunda edidición. s.l. : Prentice Hall, 2006.

Identificador: RF-002	
Nombre: Ver mensajes	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá ver en el tablón de mensajes, los mensajes públicos de los usuarios a los que sigue.	

Tabla 35. RF-002 Ver mensajes.

Identificador: RF-003	
Nombre: Ver resumen seguidos	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá ver los 10 usuarios más activos a los que sigue en la página principal.	

Tabla 36. RF-003 Ver resumen seguidos.

Identificador: RF-004	
Nombre: Ver número seguidos	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá ver el número total de usuarios a los que sigue.	

Tabla 37. RF-004 Ver número seguidos.

Identificador: RF-005	
Nombre: Ver resumen seguidores	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá ver los 10 usuarios más activos que le siguen en la página principal.	

Tabla 38. RF-005 Ver resumen seguidores.

Identificador: RF-006	
Nombre: Ver número seguidores	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá ver el número total de usuarios que le siguen.	

Tabla 39. RF-006 Ver número seguidores.

Identificador: RF-007	
Nombre: Ver todos seguidos	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá ver todos los usuarios a los que sigue.	

Tabla 40. RF-007 Ver todos seguidos.

Identificador: RF-008	
Nombre: Ver todos seguidores	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá ver todos los usuarios que le siguen.	

Tabla 41. RF-008 Ver todos seguidores.

Identificador: RF-009	
Nombre: Ver usuarios grupo	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá ver los usuarios que pertenecen a un grupo concreto cuando este viendo los mensajes de ese grupo.	

Tabla 42. RF-009 Ver usuarios grupo.

Identificador: RF-010	
Nombre: Buscar persona	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: J1 Modulo de acción de búsqueda (Sitios de búsqueda rápidos y relevantes) J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá buscar otros usuarios del sistema por nombre o email.	

Tabla 43. RF-010 Buscar persona.

Identificador: RF-011	
Nombre: Añadir seguido	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá añadir a otro usuario como seguido.	

Tabla 44. RF-011 Añadir seguido.

Identificador: RF-012	
Nombre: Eliminar seguido	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá dejar de seguir a un usuario al que ya estaba siguiendo.	

Tabla 45. RF-012 Eliminar seguido.

Identificador: RF-013	
Nombre: Bloquear seguidor	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá bloquear a un seguidor para que este no pueda ver sus mensajes.	

Tabla 46. RF-013 Bloquear seguidor.

Identificador: RF-014	
Nombre: Desbloquear seguidor	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá desbloquear a un seguidor, previamente bloqueado, para que pueda ver sus mensajes.	

Tabla 47. RF-014 Desbloquear seguidor.

Identificador: RF-015	
Nombre: Escribir mensaje	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá escribir mensajes nuevos.	

Tabla 48. RF-015 Escribir mensaje.

Identificador: RF-016	
Nombre: Escribir mensaje grupo	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá escribir mensajes nuevos, eligiendo si lo quiere publicar en el tablón de mensajes general o en un tablón de un grupo.	

Tabla 49. RF-016 Escribir mensaje grupo.

Identificador: RF-017	
Nombre: Eliminar mensaje	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá eliminar los mensajes que haya escrito.	

Tabla 50. RF-017 Eliminar mensaje.

Identificador: RF-018	
Nombre: Reescribir mensaje	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá reescribir los mensajes de otros usuarios. En este caso siempre aparecerá el autor original del mensaje.	

Tabla 51. RF-018 Reescribir mensaje.

Identificador: RF-019	
Nombre: Responder mensaje	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá responder a otro mensaje de un usuario al que siga. Esta respuesta se considerará un mensaje privado dirigido al autor del mensaje respondido, y por ello sólo este podrá verlo. Dicha respuesta se verá dentro del tablón de mensajes públicos del destinatario y del autor de esta.	

Tabla 52. RF-019 Responder mensaje.

Identificador: RF-020	
Nombre: Filtrar mensajes nombre	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: J1 Modulo de acción de búsqueda (Sitios de búsqueda rápidos y relevantes) J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá ver los mensajes de un usuario concreto de los que sigue.	

Tabla 53. RF-020 Filtrar mensajes nombre.

Identificador: RF-021	
Nombre: Filtrar mensajes fecha	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: J1 Modulo de acción de búsqueda (Sitios de búsqueda rápidos y relevantes) J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá ver los mensajes de una fecha concreta.	

Tabla 54. RF-021 Filtrar mensajes fecha.

Identificador: RF-022	
Nombre: Filtrar mensajes Intervalo	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: J1 Modulo de acción de búsqueda (Sitios de búsqueda rápidos y relevantes) J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá ver los mensajes pertenecientes a un intervalo de tiempo, es decir, los mensajes que hayan sido escritos entre dos fechas proporcionadas por el usuario.	

Tabla 55. RF-022 Filtrar mensajes intervalo.

Identificador: RF-023	
Nombre: Filtrar mensajes grupo	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: J1 Modulo de acción de búsqueda (Sitios de búsqueda rápidos y relevantes) J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá ver los mensajes de un grupo al que pertenece.	

Tabla 56. RF-023 Filtrar mensajes grupo.

Identificador: RF-024	
Nombre: Aumentar página tablón mensajes	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá incrementar el número de página del tablón para ver mensajes más antiguos. En cada página del tablón de mensajes entrarán 10 mensajes, siempre se mostrarán los más modernos, si el usuario desea ver mensajes más antiguos, incrementará en número de página.	

Tabla 57. RF-024 Aumentar página tablón mensajes.

Identificador: RF-025	
Nombre: Disminuir página tablón mensajes	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá disminuir el número de página del tablón para ver mensajes más nuevos.	

Tabla 58. RF-025 Disminuir página tablón mensajes.

Identificador: RF-026	
Nombre: Ir a página tablón mensajes	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: J3 Resultados de búsqueda organizados (Sitios de búsqueda rápidos y relevantes)	
Descripción: El usuario podrá ir a un página concreta del tablón para ver los mensajes de esa página. Se proporcionará un campo select con las páginas disponibles, de esta forma se podrá seleccionar la que se desee.	

Tabla 59. RF-026 Ir a página tablón mensajes.

Identificador: RF-027	
Nombre: Enviar privado	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá enviar un mensaje privado a otro usuario. Estos mensajes sólo podrán ser leídos por el destinatario y se mostrarán en un tablón diferente al público general.	

Tabla 60. RF-027 Enviar privado.

Identificador: RF-028	
Nombre: Ver tablón privados	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá ver la cabecera de los mensajes privados dirigidos a él, los mensajes que ya ha leído y los mensajes que ha enviado en diferentes pestañas del tablón de privados. Cada mensaje o cabecera es como una fila de una tabla en la que se podrá ver el autor del mensaje, la fecha y el asunto, no el contenido de texto del mensaje.	

Tabla 61. RF-028 Ver tablón privados.

Identificador: RF-029	
Nombre: Leer privado	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá leer el contenido de un mensaje privado concreto.	

Tabla 62. RF-029 Leer privado.

Identificador: RF-030	
Nombre: Borrar privado	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá borrar un privado seleccionado de cualquiera de las tres pestañas recibido, enviado y/o leído.	

Tabla 63. RF-030 Borrar privado.

Identificador: RF-031	
Nombre: Responder privados	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá responder a un privado. Este nuevo mensaje será otro privado que irá dirigido al autor del mensaje al que se ha respondido.	

Tabla 64. RF-031 Responder privados.

Identificador: RF-032	
Nombre: Ver respuestas	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá ver todas las respuestas a un mensaje del tablón público. Se abrirá un nuevo tablón en el que se podrán leer todas las respuestas que haya recibido un mensaje en particular.	

Tabla 65. RF-032 Ver respuestas.

Identificador: RF-033	
Nombre: Ver mensaje respuesta	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá ver el mensaje al que responde otro recibido o a que mensaje ha respondido, desde el propio mensaje de respuesta.	

Tabla 66. RF-033 Ver mensaje respuesta.

Identificador: RF-034	
Nombre: Usuario activo	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá ver el nombre de usuario de la sesión con el que se conecto.	

Tabla 67. RF-034 Usuario activo.

Identificador: RF-035	
Nombre: Salir del sistema	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá desconectarse del sistema.	

Tabla 68. RF-035 Salir del sistema.

Identificador: RF-036	
Nombre: Actualización automática	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Se actualizará el tablón de mensajes, el tablón de contactos (seguidos y seguidores) y el tablón de privados automáticamente cada 25 segundos, pero sólo en el caso de que hubiera alguna modificación que ver por parte del usuario autenticado.	

Tabla 69. RF-036 Actualización automática.

Identificador: RF-037	
Nombre: Contenido personalizado	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Cuando el usuario se autentique en el sistema se le ofrecerá contenido personalizado, más concretamente los grupos de asignaturas que tenga matriculadas cada usuario, de esta forma se podrán ver los mensajes agrupados por asignaturas.	

Tabla 70. RF-037 Contenido personalizado.

Requisitos no funcionales

Requisitos de sistema

Identificador: RS-001	
Nombre: Directorio	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El sistema deberá usar un servidor de directorios donde estarán almacenados los datos de los usuarios del sistema. Los usuarios se validarán sobre este directorio.	

Tabla 71. RS-001 Directorio.

Identificador: RS-002	
Nombre: Compatibilidad navegadores web	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La aplicación deberá ser compatible con Internet Explorer 7.0, Mozilla Firefox 7.0 y versiones superiores como mínimo.	

Tabla 72. RS-002 Compatibilidad navegadores web.

Identificador: RS-003	
Nombre: Cumplimiento de estándares	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La aplicación cumplirá los estándares de calidad de código CSS 3 de la organización W3C.	

Tabla 73. RS-003 Cumplimiento de estándares.

Identificador: RS-004	
Nombre: BBDD	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Se utilizará MySql 5.1 como sistema gestor de bases de datos.	

Tabla 74. RS-004 BBDD.

Identificador: RS-005	
Nombre: Tecnología de desarrollo	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El sistema será desarrollado utilizando tecnologías Ajax (HTML, JavaScript, CSS).	

Tabla 75. RS-005 Tecnología de desarrollo.

Identificador: RS-006	
Nombre: Lenguaje servidor	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El código del lado del servidor será desarrollado en Java con Servlet 3.0 y JSP 2.2. Se utilizará el jdk 1.6.0 para el desarrollo y la ejecución del programa.	

Tabla 76. RS-006 Lenguaje servidor.

Identificador: RS-007	
Nombre: Librería Interfaz	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Para desarrollar la interfaz de usuario, se utilizarán la librerías JQuery v1.4.2 y JQuery UI v1.8.5	

Tabla 77. RS-007 Librería interfaz.

Identificador: RS-008	
Nombre: Librería Ajax	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Para realizar la parte de Ajax se utilizará la librería DWR v2.0.1.	

Tabla 78. RS-008 Librería Ajax.

Identificador: RS-009	
Nombre: IDE	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El desarrollo de la aplicación se realizará con la ayuda de Eclipse Helios.	

Tabla 79. RS-009 IDE.

Requisitos de seguridad

Identificador: RSG-001	
Nombre: Sesión	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: A10 Sistema de Autenticación (Aplicaciones web que trabajan)	
Descripción: Cada vez que un usuario se valide en la aplicación se creará una sesión nueva con las variables de usuario. De esta forma se conseguirá que no se pueda acceder a la aplicación sin identificarse, y que el usuario este identificado. La sesión se cerrará cuando el usuario salga del sistema o cuando esta caduque tras 30 minutos.	

Tabla 80. RSG-001 Sesión.

Identificador: RSG-002	
Nombre: Contraseñas cifradas	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Las contraseñas almacenadas en el directorio LDAP estarán cifradas mediante el formato SHA.	

Tabla 81. RSG-002 Contraseñas cifradas.

Requisitos de usabilidad

Identificador: RU-001	
Nombre: Mensajes de operación	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: A10 Ofrecer feedback informativo (Aplicaciones web que trabajan)	
Descripción: Cada vez que el usuario realice una operación sobre el sistema, este devolverá un mensaje para avisarle del resultado de dicha operación. Por ejemplo al autenticarse, al enviar un mensaje nuevo, o al añadir un nuevo usuario como seguido.	

Tabla 82. RU-001 Mensajes de operación.

Identificador: RU-002	
Nombre: Estilo mensajes	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Se diferenciaran los tipos de mensajes para una clara, rápida e intuitiva visión de estos. Se hará mediante diferente tipo de letra, cursiva, colores, etc.	

Tabla 83. RU-002 Estilo mensajes.

Identificador: RU-003	
Nombre: Ayuda escritura mensajes color	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☐ MEDIA ☑ BAJA
Patrón: N/A	
Descripción: El mensaje que escribe el usuario irá cambiando de color dependiendo de cuantos caracteres lleve escritos, de esta forma el usuario tendrá una ayuda visual a la hora de añadir nuevos mensajes.	

Tabla 84. RU-003 Ayuda escritura mensajes color.

Identificador: RU-004	
Nombre: Ayuda escritura mensajes número caracteres.	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Mientras se escribe un nuevo mensaje se mostrará el número de caracteres introducidos en cada momento.	

Tabla 85. RU-004 Ayuda escritura mensajes número caracteres.

Identificador: RU-005	
Nombre: Estilo usuarios bloqueados.	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Se mostrarán de color diferente los seguidores que hayan sido bloqueados.	

Tabla 86. RU-005 Estilo usuarios bloqueados.

Identificador: RU-006	
Nombre: Estilo URL's	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: K10 Links Obvios(Haciendo la navegación fácil)	
Descripción: Las URL de los mensajes que las contengan se mostrarán como enlaces para que el usuario pueda acceder pinchando sobre ellos. Además se le dará un color diferente dependiendo si el enlace ha sido visitado o no.	

Tabla 87. RU-006 URL's

Identificador: RU-007	
Nombre: Mensaje confirmación borrado.	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Cuando el usuario quiera borrar un mensaje o un privado aparecerá una ventana para confirmar el borrado.	

Tabla 88. RU-007 Mensaje confirmación borrado.

Identificador: RU-008	
Nombre: Estilo botón privados	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El botón de ver privados cambiará dependiendo de los privados recibidos sin leer, de esta forma el usuario tendrá una forma fácil de saber si tiene privados o no. El texto del botón cambiara de color cuando haya algún nuevo privado y a su vez aparecerá un número que indicará cuantos nuevos privados hay.	

Tabla 89. RU-008 Estilo botón privados.

Identificador: RU-009	
Nombre: Estilo tablón respuestas	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El mensaje principal del que se quieren ver las respuestas tendrá un estilo diferente que destacará para hacerlo más intuitivo. Este requisito se refiere a los mensajes del tablón de respuestas.	

Tabla 90. RU-009 Estilo tablón respuestas.

Identificador: RU-010	
Nombre: Instrucciones	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: A10 Ayuda de uso al usuario mediante mensajes (Aplicaciones web que trabajan) H8 Ayuda sensible al contexto (Ayudando a los usuarios a completar tareas)	
Descripción: Manteniendo el ratón sobre los elementos de la interfaz aparecerán instrucciones sobre para que sirve cada uno, de esta forma el aprendizaje del usuario es más dinámico que con un texto explicando todo el funcionamiento. Además el usuario no pierde el contexto de lo que está haciendo.	

Tabla 91. RU-010 Instrucciones.

Identificador: RU-011	
Nombre: Maquetación líquida	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La maquetación de la interfaz será líquida con respecto a la resolución y al tamaño de fuente.	

Tabla 92. RU-011 Maquetación líquida.

Identificador: RU-012	
Nombre: Ver número página tablón mensajes	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: El usuario podrá ver el número de página del tablón de mensajes que está viendo.	

Tabla 93. RU-012 Ver número página tablón mensajes.

Identificador: RU-013	
Nombre: Ventanas modales	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: H1 Proceso de embudo (Ayudando a los usuarios a completar tareas) H6 Ventanas flotantes (Ayudando a los usuarios a completar tareas)	
Descripción: Para ayudar a completar actividades que no están en la pantalla principal, como ver, y enviar privados, borrar mensajes, etc. se realizarán en ventanas modales que mantienen el foco en la actividad que se está realizando, y a su vez podrá realizar la tarea sin perder el contexto.	

Tabla 94. RU-013 Ventanas modales.

Identificador: RU-014	
Nombre: Mensajes de error	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: H1 Proceso de embudo (Ayudando a los usuarios a completar tareas) K13 Mensajes de error útiles (Haciendo la navegación fácil)	
Descripción: Cuando se produzca un error en una operación se mostrará con un mensaje, de esta forma el usuario sabrá como corregir el error, y se le ayuda a completar la tarea.	

Tabla 95. RU-014 Mensajes de error.

Identificador: RU-015	
Nombre: Links externos	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: K8 Links externos (Haciendo la navegación fácil)	
Descripción: Los enlaces a páginas externas se abrirán en una ventana o pestaña nueva para evitar confusiones a los usuarios al perder el contexto.	

Tabla 96. RU-015 Links externos.

Identificador: RU-016	
Nombre: Lenguaje familiar	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: K11 Lenguaje familiar (Haciendo la navegación fácil)	
Descripción: En toda la aplicación se utilizará un lenguaje familiar que todos los usuarios entiendan.	

Tabla 97. RU-016 Lenguaje familiar.

Identificador: RU-017	
Nombre: Prevención de errores	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: K12 Previniendo errores (Haciendo la navegación fácil)	
Descripción: Se procurará evitar los errores de los usuarios todo lo posible. Por ejemplo las fechas del formulario de búsqueda de mensajes serán datepickers, de forma que no puedan equivocar el formato, o el select de cambio de página en el tablón de mensajes, donde se le darán sólo las opciones de las páginas disponibles.	

Tabla 98. RU-017 Prevención de errores.

Identificador: RU-018	
Nombre: Hilo respuestas privados	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: Cuando el usuario lea el contenido de un privado, si existen respuestas a ese privado también se mostrarán, pudiéndose ver de esa manera todo el hilo de respuestas del privado, y siguiendo mucho mejor la conversación.	

Tabla 99. RU-018 Hilo respuestas privados.

Requisitos de interfaz

Identificador: RI-001	
Nombre: Formulario autenticación	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un formulario de autenticación con un campo de texto para el nombre de usuario y otro para la contraseña.	

Tabla 100. RI-001 Formulario autenticación.

Identificador: RI-002	
Nombre: Formulario nuevo mensaje	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un formulario que consistirá en un textarea para introducir el texto del mensaje y un select que permita seleccionar el tablón de grupo donde se quiere publicar el mensaje. Este campo select se rellenará automáticamente con los grupos que tenga cada usuario.	

Tabla 101. RI-002 Formulario nuevo mensaje.

Identificador: RI-003	
Nombre: Formulario buscar persona	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un formulario de búsqueda con un campo de texto y un select con dos opciones para poder buscar por nombre o email.	

Tabla 102. RI-003 Formulario buscar persona.

Identificador: RI-004	
Nombre: Botón eliminar seguido	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón en el tablón de contactos de seguidos para dejar de seguir a un usuario.	

Tabla 103. RI-004 Botón eliminar seguido.

Identificador: RI-005	
Nombre: Botón bloquear	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón en el tablón de contactos de seguidores para bloquear a un seguidor.	

Tabla 104. RI-005 Botón bloquear.

Identificador: RI-006	
Nombre: Botón eliminar mensaje	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón en cada mensaje para poder borrarlo. Este botón solo debe aparecer en los mensajes propios del usuario que está conectado, de esta forma sólo podrá borrar sus mensajes.	

Tabla 105. RI-006 Botón eliminar mensaje.

Identificador: RI-007	
Nombre: Botón reescribir mensaje	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón en cada mensaje de otro usuario para poder reescribir el mensaje. No aparecerá en los mensajes propios de cada usuario para que no puedan reescribir sus propios mensajes.	

Tabla 106. RI-007 Botón reescribir mensaje.

Identificador: RI-008	
Nombre: Botón responder mensaje	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón en cada mensaje de otro usuario para poder responder a dichos mensajes. No aparecerá en los mensajes propios de cada usuario para que no puedan responder sus propios mensajes.	

Tabla 107. RI-008 Botón responder mensaje.

Identificador: RI-009	
Nombre: Formulario de respuesta	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un formulario con un textarea para responder a un mensaje. El formulario aparecerá cuando se pulse el botón de responder mensaje.	

Tabla 108. RI-009 Formulario de respuesta.

Identificador: RI-010	
Nombre: Filtro Select usuario	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un campo select de usuarios para poder filtrar los mensajes del tablón por un usuario. Este campo select será rellenado automáticamente con los usuarios que está siguiendo cada usuario.	

Tabla 109. RI-010 Select usuario.

Identificador: RI-011	
Nombre: Filtro Select grupo	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un select de grupos para poder filtrar los mensajes del tablón por un grupo. Este campo select será rellenado automáticamente con los grupos de cada usuario.	

Tabla 110. RI-011 Filtro Select grupo.

Identificador: RI-012	
Nombre: Filtro Input fechas	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener dos inputs donde se le indicaran dos fechas para realizar el filtro por fecha, tanto de día como de intervalo de tiempo.	

Tabla 111. RI-012 Filtro Input fechas.

Identificador: RI-013	
Nombre: Botones paginación	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener una barra con botones que permitan avanzar a otra página, retroceder página e ir a la primera página del tablón de mensajes.	

Tabla 112. RI-013 Botones paginación.

Identificador: RI-014	
Nombre: Select paginación	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un select que permita ir directamente a una página concreta del tablón de mensajes. Este campo se rellenará automáticamente con las páginas disponibles según el número de mensajes que haya en el tablón.	

Tabla 113. RI-014 Select paginación.

Identificador: RI-015	
Nombre: Tabla privados	
Necesidad: ☐ ESENCIAL ☐ DESEABLE ☑ OPCIONAL	**Prioridad:** ☐ ALTA ☐ MEDIA ☑ BAJA
Patrón: K3 Tabla de pestañas (Haciendo la navegación fácil)	
Descripción: La interfaz deberá tener un elemento donde se mostrarán las cabeceras de los mensajes privados recibidos, leídos y enviados. Cada privado se verá como una fila de una tabla. La forma de mostrar este contenido será mediante un tablón de pestañas ya que facilita en gran medida su visualización.	

Tabla 114. RI-015 Tabla privados.

Identificador: RI-016	
Nombre: Botón enviar privado	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón disponible para enviar un privado a un usuario.	

Tabla 115. RI-016 Botón enviar privado.

Identificador: RI-017	
Nombre: Formulario privado	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un formulario con dos campos de texto para introducir el asunto y el texto del mensaje privado.	

Tabla 116. RI-017 Formulario privado.

Identificador: RI-018	
Nombre: Botón leer privado	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón para poder leer un mensaje privado.	

Tabla 117. RI-018 Botón leer privado.

Identificador: RI-019	
Nombre: Botón responder privado	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón para poder responder a un mensaje privado.	

Tabla 118. RI-019 Botón responder privado.

Identificador: RI-020	
Nombre: Botón eliminar privado	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón para poder eliminar un mensaje privado.	

Tabla 119. RI-020 Botón eliminar privado.

Identificador: RI-021	
Nombre: Tablón privados	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un tablón donde se mostrará el texto de los privados que se quieran leer. Este requisito es diferente al RI-015 Tabla privados, ya que en el otro se muestran las cabeceras de los privados y en este su contenido de texto.	

Tabla 120. RI-021 Tablón privados.

Identificador: RI-022	
Nombre: Botón ver privados	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón para ver los privados. Estará en la barra superior y cambiará de estilo dependiendo de si hay nuevos privados sin leer o no. Relacionado con el requisito RU-008 Estilo botón privados.	

Tabla 121. RI-022 Botón ver privados.

Identificador: RI-023	
Nombre: Botón ver respuestas	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz tendrá un botón en cada mensaje para poder ver todas las respuestas a ese mensaje.	

Tabla 122. RI-023 Botón ver respuestas.

Identificador: RI-024	
Nombre: Tablón respuestas	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz tendrá un elemento donde se mostrarán las respuestas. Estará oculto y se mostrará cuando se pinche sobre el botón de ver respuestas. Aparecerá fuera de la vista del tablón principal para que el usuario no pierda el contexto del mensaje del que está observando las respuestas.	

Tabla 123. RI-024 Tablón respuestas.

Identificador: RI-025	
Nombre: Enlace mensaje respuesta	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un enlace en cada mensaje de respuesta, que permitirá ver el mensaje origen al que responde.	

Tabla 124. RI-025 Enlace mensaje respuesta.

Identificador: RI-026	
Nombre: Ventana respuesta	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener una ventana donde se mostrará el mensaje al que responde otro.	

Tabla 125. RI-026 Ventana respuesta.

Identificador: RI-027	
Nombre: Botón salir	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un botón para poder salir de la aplicación.	

Tabla 126. RI-027 Botón salir.

Identificador: RI-028	
Nombre: Elemento usuario	
Necesidad: ☐ ESENCIAL ☑ DESEABLE ☐ OPCIONAL	**Prioridad:** ☐ ALTA ☑ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz deberá tener un elemento que muestre el nombre del usuario identificado en la sesión, en la parte superior derecha.	

Tabla 127. RI-028 Elemento usuario.

Identificador: RI-029	
Nombre: Página única	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz del sistema estará formada por dos únicas páginas. La página de autenticación, y la página principal. En la página principal se realizarán todas las funciones y se actualizará dinámicamente. El objetivo de este requisito es que la aplicación sea más sencilla e intuitiva para el usuario, ya que todas las operaciones estarán disponibles desde la misma página y el usuario mantendrá el contexto en todo momento.	

Tabla 128. RI-029 Página única.

Identificador: RI-030	
Nombre: Interfaz Ajax	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: A10 Aplicaciones web que trabajan (Géneros del sitio)	
Descripción: Se implementará una interfaz Ajax (con botones de acción, tablones que serán rellenados dinámicamente, etc.) que proporcionará control directo a los usuarios.	

Tabla 129. RI-030 Interfaz Ajax.

Identificador: RI-031	
Nombre: Barra de navegación	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: K2 Barra de navegación (Haciendo la navegación fácil)	
Descripción: La interfaz tendrá en la parte superior una barra donde se incluirán botones para operaciones especiales, como ver el tablón de privados o buscar usuarios, de esta forma el usuario tendrá más fácil acceder a estas actividades.	

Tabla 130. RI-031 Barra de navegación.

Identificador: RI-032	
Nombre: Barra lateral	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: K2 Barra de navegación (Haciendo la navegación fácil)	
Descripción: La interfaz tendrá en la parte derecha una barra lateral en la que se incluirá el tablón de contactos y un campo textarea con el que se introducirán nuevos mensajes.	

Tabla 131. RI-032 Barra lateral.

Identificador: RI-033	
Nombre: Botones de acción	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: K4 Botones de acción (Haciendo la navegación fácil)	
Descripción: Se usarán botones para representar las acciones que tendrán un diseño gráfico claro para que se distingan bien que son botones que se pueden pulsar. Además las etiquetas serán claras y en el caso de sólo ser iconos se presentará información adicional.	

Tabla 132. RI-033 Botones de acción.

Identificador: RI-034	
Nombre: Botón seguir usuario	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz tendrá un botón para seguir a un nuevo usuario.	

Tabla 133. RI-034 Botón seguir usuario.

Identificador: RI-035	
Nombre: Botón ver seguidos	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz tendrá un botón para ver los usuarios a los que se está siguiendo.	

Tabla 134. RI-035 Botón ver seguidos.

Identificador: RI-036	
Nombre: Botón ver seguidores	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz tendrá un botón para ver los usuarios seguidores.	

Tabla 135. RI-036 Botón ver seguidores.

Identificador: RI-037	
Nombre: Reloj de sesión	
Necesidad: ☑ ESENCIAL ☐ DESEABLE ☐ OPCIONAL	**Prioridad:** ☑ ALTA ☐ MEDIA ☐ BAJA
Patrón: N/A	
Descripción: La interfaz tendrá un reloj de cuenta atrás, que indicará al usuario cuanto tiempo le queda antes de que se cierre la sesión.	

Tabla 136. RI-037 Reloj de sesión.

Matriz Requisitos Software - Casos de uso

A continuación se exponen las tablas de trazabilidad entre requisitos y casos de uso. Estas tablas representan la relación que hay entre los requisitos obtenidos y los casos de uso. La primera tabla relaciona los requisitos funcionales con los casos de uso, la segunda los de usabilidad y la tercera los de interfaz.

		CASOS DE USO																			
		CU-01	CU-02	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10	CU-11	CU-12	CU-13	CU-14	CU-15	CU-16	CU-17	CU-18	CU-19	CU-20
REQUISITOS SOFTWARE FUNCIONALES	RF-001	X																			
	RF-002			X																	
	RF-003											X									
	RF-004											X									
	RF-005												X								
	RF-006												X								
	RF-007											X									
	RF-008												X								
	RF-009											X									
	RF-010						X														
	RF-011							X													
	RF-012								X												
	RF-013									X											
	RF-014										X										
	RF-015													X							
	RF-016													X							
	RF-017														X						
	RF-018															X					
	RF-019																X				
	RF-020				X																
	RF-021				X																
	RF-022				X																
	RF-023				X																
	RF-024					X															
	RF-025					X															
	RF-026					X															
	RF-027																	X			
	RF-028																		X	X	X
	RF-029																		X		
	RF-030																			X	
	RF-031																				X
	RF-032			X																	
	RF-033			X																	
	RF-034	X																			
	RF-035		X																		
	RF-036																				
	RF-037			X										X							

		CASOS DE USO																			
		CU-01	CU-02	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10	CU-11	CU-12	CU-13	CU-14	CU-15	CU-16	CU-17	CU-18	CU-19	CU-20
REQUISITOS SOFTWARE USABILIDAD	RU-001	X						X						X							
	RU-002			X																	
	RU-003													X							
	RU-004													X							
	RU-005									X	X		X								
	RU-006			X																	
	RU-007														X					X	
	RU-008																		X	X	X
	RU-009			X																	
	RU-010				X				X	X	X	X	X		X	X	X	X			
	RU-011																				
	RU-012					X															
	RU-013						X	X				X	X				X	X	X	X	X
	RU-014	X					X	X						X			X	X			X
	RU-015			X																	
	RU-016	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	RU-017				X	X	X														
	RU-018																		X		

		CASOS DE USO																			
		CU-01	CU-02	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10	CU-11	CU-12	CU-13	CU-14	CU-15	CU-16	CU-17	CU-18	CU-19	CU-20
REQUISITOS SOFTWARE INTERFAZ	RI-001	X																			
	RI-002													X							
	RI-003						X														
	RI-004								X												
	RI-005									X	X										
	RI-006														X						
	RI-007															X					
	RI-008																X				
	RI-009																X				
	RI-010				X																
	RI-011				X																
	RI-012				X																
	RI-013					X															
	RI-014					X															
	RI-015																		X	X	X
	RI-016																	X			
	RI-017																	X			
	RI-018																		X		
	RI-019																				X
	RI-020																			X	
	RI-021																		X	X	X
	RI-022																		X	X	X
	RI-023			X																	
	RI-024			X																	
	RI-025			X													X				
	RI-026			X													X				
	RI-027		X																		
	RI-028																				
	RI-029																				
	RI-030																				
	RI-031																				
	RI-032																				
	RI-033																				
	RI-034							X													
	RI-035											X									
	RI-036												X								
	RI-037	X																			

Apéndice B: Casos de uso.

Introducción

Propósito

El objetivo de este documento es expresar los casos de uso del sistema de Microblogging a desarrollar, tanto los diagramas de casos de uso como los casos de uso extendidos.

Personal involucrado

Nombre	Ignacio Vidal Hernando
Rol	Jefe de Proyecto, Analista, Diseñador, Desarrollador.
Categoría profesional	Programador Junior.
Responsabilidades	Se encargará de todo lo relacionado con el proyecto. Recogida y análisis de requisitos, diseño del sistema, desarrollo, pruebas, etc.

Alcance

Los casos de uso describen las funcionalidades deseadas del sistema que se va a desarrollar. De cada uno se expondrán una descripción, los actores involucrados, una precondición y una poscondición, y los posibles escenarios, de éxito o alternativos.

Acrónimos y definiciones

Acrónimos

- **BD:** Base de datos.
- **ID:** Identificador.
- **HTML:** HyperText Markup Language.

Definiciones

- **Select:** Campo de entrada HTML de una interfaz web en la que se proporcionan una serie de opciones para seleccionar.
- **Tablón de usuarios:** Es una parte de la interfaz de la aplicación donde aparecerán los usuarios seguidos y seguidores.
- **Tablón de mensajes:** Es una parte de la interfaz de la aplicación donde se mostrarán los mensajes públicos de los usuarios.
- **Tablón de privados:** Es una parte de la interfaz de la aplicación donde se mostrarán los mensajes privados de los usuarios. Estará dividido en tres partes para ver los privados recibidos, leídos y enviados. Tendrá otra división donde se mostrará el contenido de los privados cuando el usuario seleccione alguno, ya que en principio solo se verá una cabecera con el usuario el asunto y la fecha de los privados.
- **Tablón de búsqueda de usuarios:** Es una parte de la interfaz de la aplicación donde se mostrarán los usuarios que resulten de la actividad buscar usuarios.

Resumen

En el resto del documento se van a exponer los casos de uso que representarán de forma general la funcionalidad del sistema. Se definirá la información sobre los actores que participan, los diagramas de caso de uso, y finalmente se expondrán de forma detallada los casos de uso en formato extendido.

Actores del sistema

Diagrama de actores

Este diagrama representa los actores que participan en el sistema.

Ilustración 47.Diagrama de actores.

Detalles de actores

Actor	Descripción
Usuario	Es un usuario que estará dentro del directorio. Da igual si ya es usuario registrado del sistema microblogging o no, ya que el propio sistema se encarga de meterlo en la base de datos de la aplicación en caso de que sea su primer acceso. Los usuarios que no estén en el directorio no podrán hacer uso del sistema.

Tabla 137. Detalles de los actores.

Diagramas de casos de uso

En este apartado se definen los diagramas de casos de uso generados. Habrá un diagrama en general que ponga todos en contexto y a continuación se definirán algunos diagramas de más bajo nivel que permitirán ver los casos de uso de forma detallada. En el primer diagrama se pueden observar los casos de uso generales, en el que se ponen en contexto todos ellos. En el segundo, se expande el caso de uso de visualizar mensajes para ver de forma más detallada los casos de uso de más bajo nivel.

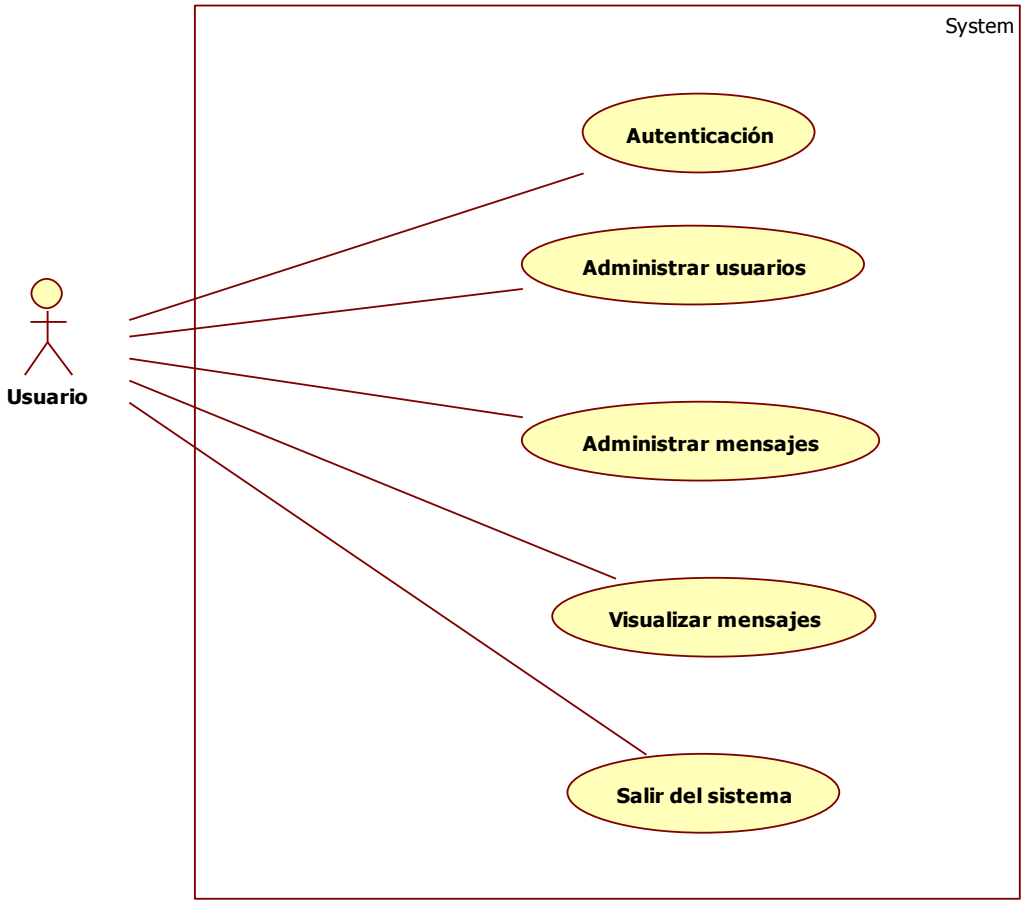


Ilustración 48.Diagrama de casos de uso general.

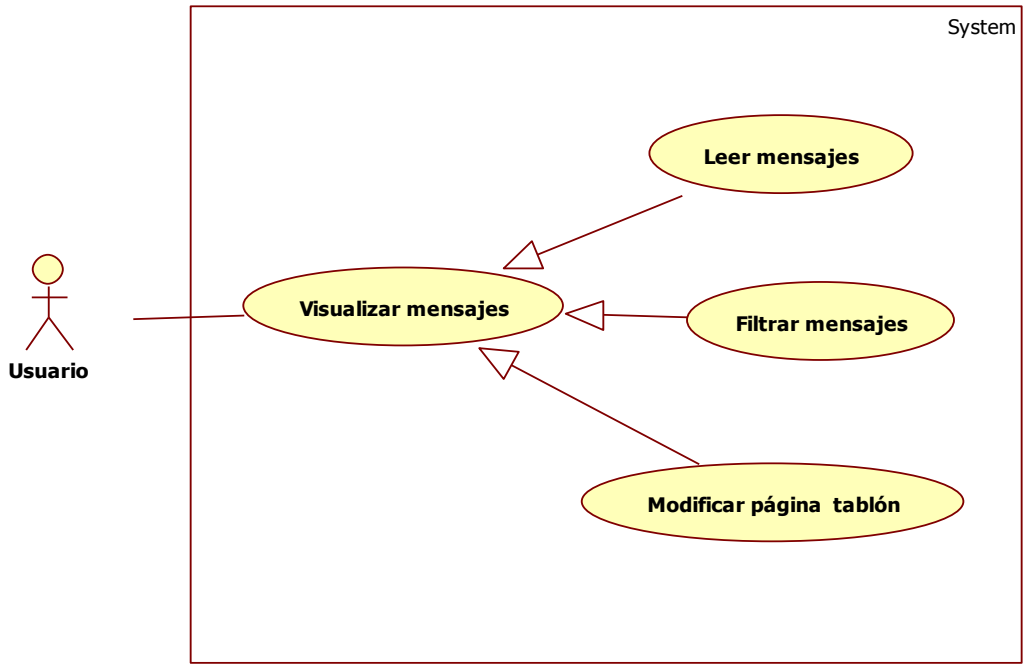


Ilustración 49.Diagrama de casos de uso "Visualizar mensajes".

En el siguiente diagrama se puede observar como el caso de uso administrar mensajes extiende a otros dos que a su vez agrupan otros casos de uso de más bajo nivel.

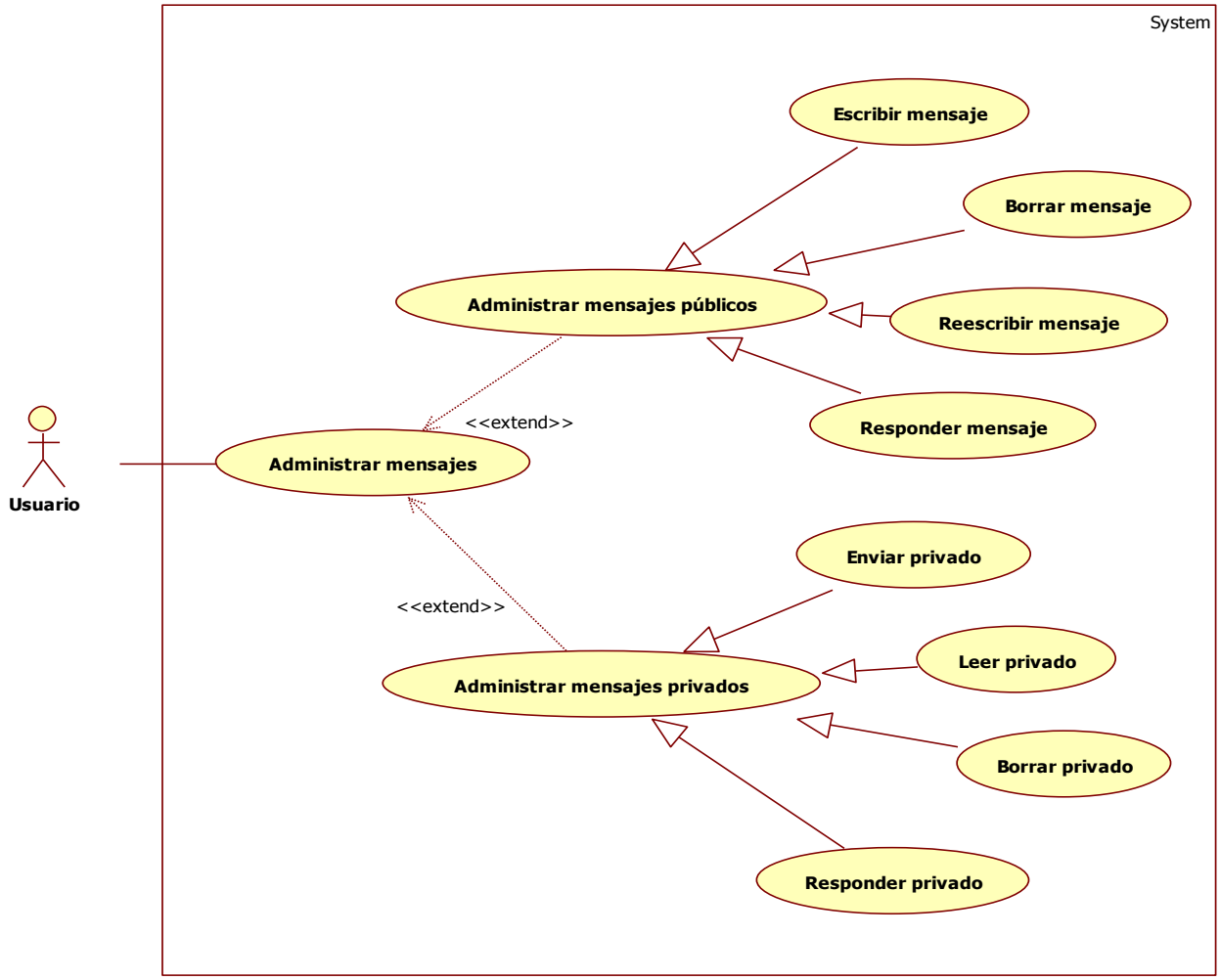


Ilustración 50.Diagrama de casos de uso "Administrar mensajes"

En el último diagrama se observa la expansión del caso de uso Administrar usuarios.

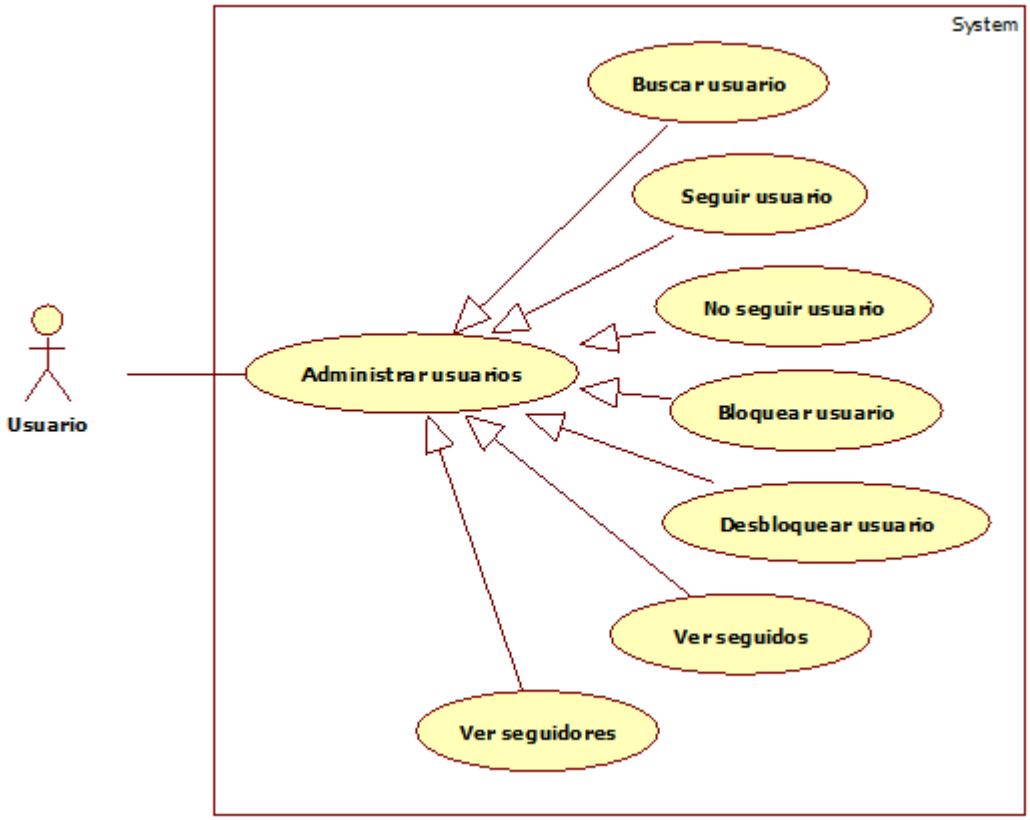


Ilustración 51.Diagrama de casos de uso "Administrar usuarios".

Casos de uso en formato extendido

En este apartado se definirán los casos de uso en formato extendido que aparecen en los diagramas. Cada caso de uso extendido será una tabla con los siguientes atributos:

- **Caso de uso:** Será el nombre del caso de uso que se extiende.
- **Identificador:** Es un campo alfanumérico que identificará al caso de uso de forma única.
- **Actor:** Es el actor que participa en el caso de uso.
- **Descripción:** Texto que explica de forma simple la función del caso de uso.
- **Precondición:** Son los prerrequisitos que son necesarios para que ocurra dicho caso de uso.
- **Escenario normal:** Define la interacción entre usuario y sistema que se produce en el caso de que no haya ningún error.
- **Escenario alternativo:** Define la interacción entre usuario y sistema que se produce en el caso de que haya algún error.
- **Poscondición:** Define como queda el sistema una vez se ha producido el caso de uso.

CASO DE USO	Autenticación	
IDENTIFICADOR	CU-01	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere autenticarse para utilizar la aplicación.	
PRECONDICIÓN	El usuario no debe estar autenticado.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. Introduce su nombre y clave. 2. Envía los datos al sistema.	3. Comprueba los datos obtenidos sobre el directorio. 4. Comprueba si es la primera conexión al sistema. - Si es la primera, copia los datos necesarios en la base de datos de la aplicación. Sigue a 5. - Si no es la primera pasa a 5. 5. Inicia una nueva sesión para el usuario. 6. Muestra la interfaz principal. 7. Muestra un elemento con el nombre del usuario autenticado.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
		3. Los datos introducidos en el paso 1 son incorrectos 4. Envía un mensaje al usuario indicándole que el nombre o la clave son incorrectos.
POSTCONDICIÓN	Se crea una nueva sesión para el usuario que queda conectado en el sistema y puede realizar todas las funcionalidades.	

Tabla 138. Caso de uso "Autenticación"

CASO DE USO	Salir del sistema	
IDENTIFICADOR	CU-02	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere salir de la aplicación.	
PRECONDICIÓN	El usuario debe estar autenticado.	
ESCENARIO NORMAL	ACTOR	SISTEMA
	1. El usuario pincha sobre el botón de salir.	2. Cierra la sesión y devuelve al usuario a la página de autenticación.
ESCENARIO ALTERNATIVO	ACTOR	SISTEMA
	--	--
POSTCONDICIÓN	Se cierra la sesión del usuario. El usuario queda en la página de autenticación.	

Tabla 139. Caso de uso "Salir del sistema".

CASO DE USO	Leer mensajes	
IDENTIFICADOR	CU-03	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario desea leer los mensajes públicos del tablón.	
PRECONDICIÓN	El usuario debe estar autenticado.	
ESCENARIO NORMAL	ACTOR	SISTEMA
		1. Obtiene los usuarios que está siguiendo el usuario. 2. Obtiene los 10 mensajes más actuales de los usuarios correspondientes del punto 1. 3. Muestra los mensajes en el tablón público.
ESCENARIO ALTERNATIVO	ACTOR	SISTEMA
	--	--
POSTCONDICIÓN	El usuario estará en la página principal. El tablón mostrará los 10 mensajes más actuales de los usuarios a los que sigue.	

Tabla 140. Caso de uso "Leer mensajes".

CASO DE USO	Filtrar mensajes	
IDENTIFICADOR	CU-04	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere ver mensajes filtrados por alguna de las opciones de filtrado. Estas opciones son usuario, grupo, fecha e intervalo de fechas. Además puede mezclar las opciones para hacer filtros más específicos.	
PRECONDICIÓN	El usuario debe estar autenticado. El tablón debe haberse generado.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario introduce los datos de filtro que desee. 2. Envía los datos al sistema.	3. Comprueba los datos obtenidos para saber qué tipo de búsqueda debe realizar. 4. Realiza la búsqueda de los mensajes con los datos obtenidos, y obtiene los 10 mensajes más actuales. 5. Muestra los mensajes en el tablón público.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
		4. Realiza la búsqueda de los mensajes con los datos obtenidos, pero no hay mensajes que cumplan los requisitos del filtro. 5. No muestra ningún mensaje, quedando el tablón vacio.
POSTCONDICIÓN	El usuario estará en la página principal. El tablón mostrará los 10 mensajes más actuales que cumplan con los datos del filtro de búsqueda.	

Tabla 141. Caso de uso "Filtrar mensajes".

CASO DE USO	Modificar página tablón.	
IDENTIFICADOR	CU-05	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere ver otra página del tablón de mensajes.	
PRECONDICIÓN	El usuario debe estar autenticado. El tablón debe haberse generado. Debe haber más de 10 mensajes en el tablón.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario selecciona la página que desea ver mediante un campo select de páginas o usando los botones de navegación secuencial.	2. Recoge el número de página a mostrar. 3. Obtiene los mensajes correspondientes a la página que tiene que buscar. 4. Muestra los mensajes en el tablón público. 5. Dependiendo del número de página genera dinámicamente el formulario de selección de página del tablón.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	--	--
POSTCONDICIÓN	El usuario estará en la página principal. El tablón mostrará los mensajes de la página correspondiente.	

Tabla 142. Caso de uso "Modificar página tablón".

CASO DE USO	Buscar usuario	
IDENTIFICADOR	CU-06	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere buscar a otro en el sistema.	
PRECONDICIÓN	El usuario debe estar autenticado.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario introduce el nombre o el email del usuario al que quiere buscar. 2. Envía los datos al sistema.	3. Realiza la búsqueda con los datos pasados por el usuario. 4. Muestra en una tabla los resultados obtenidos de la búsqueda, creando también los botones de seguir usuario para cada resultado.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
		4. La búsqueda no ha encontrado ninguna coincidencia, por lo que escribe un mensaje informando de ello.
POSTCONDICIÓN	El usuario verá los usuarios que coincidan con sus criterios de búsqueda en la tabla de resultados.	

Tabla 143. Caso de uso "Buscar usuario".

CASO DE USO	Seguir usuario	
IDENTIFICADOR	CU-07	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere seguir a otro usuario del sistema para poder ver sus mensajes.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario debe haber buscado algún usuario y haber encontrado alguna coincidencia, de modo que pueda ver al usuario que quiere seguir en el tablón de búsqueda de usuarios.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de seguir usuario.	2. Comprueba si el usuario pasado ya está siendo seguido. 3. Añade el usuario como seguido del usuario autenticado. 4. Actualiza el tablón de mensajes, y el tablón de usuarios. 5. Envía un mensaje al usuario de operación correcta.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
		2. Comprueba si el usuario pasado ya está siendo seguido. 3. El usuario ya está siendo seguido, por lo que envía un mensaje indicándoselo al usuario.
POSTCONDICIÓN	El usuario tendrá un nuevo usuario seguido, pudiendo ver sus mensajes. En el tablón de usuarios aparecerá el nuevo usuario seguido.	

Tabla 144. Caso de uso "Seguir usuario".

CASO DE USO	No seguir usuario	
IDENTIFICADOR	CU-08	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere dejar de seguir a un usuario porque le han dejado de interesar sus mensajes.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario que se quiere dejar de seguir debe ser un usuario seguido.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de dejar de seguir en el tablón de usuarios seguidos.	2. Elimina el usuario que se quiere dejar de seguir de la lista de usuarios seguidos del usuario que realiza la acción. 3. Actualiza el tablón de mensajes, y el tablón de usuarios.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	--	--
POSTCONDICIÓN	El usuario verá el tablón de mensajes pero sin los mensajes del usuario al que se ha dejado de seguir. En el tablón de usuarios no aparecerá el usuario al que se ha dejado de seguir.	

Tabla 145. Caso de uso "No seguir usuario"

CASO DE USO	Bloquear usuario	
IDENTIFICADOR	CU-09	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere bloquear a un seguidor para que no pueda ver sus mensajes.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario al que se quiere bloquear debe estar siguiendo al usuario que realiza la acción.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de bloquear en el tablón de usuarios seguidores.	2. Añade a la tabla de bloqueados el usuario pasado. 3. Modifica el estilo del usuario en la tabla de seguidores para indicar que se ha bloqueado, añadiendo también el botón de desbloquear.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	--	--
POSTCONDICIÓN	El usuario verá el tablón de seguidores con el usuario bloqueado con otro estilo. Al lado del usuario bloqueado habrá un botón de desbloquear.	

Tabla 146. Caso de uso "Bloquear usuario".

CASO DE USO	Desbloquear usuario	
IDENTIFICADOR	CU-10	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere desbloquear a un seguidor para que pueda volver a ver sus mensajes.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario que se quiere desbloquear debe estar bloqueado por el usuario que realiza la acción.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de desbloquear en el tablón de usuarios seguidores.	2. Borra de la tabla de bloqueados el usuario pasado. 3. Modifica el estilo del usuario en la tabla de seguidores para indicar que ya no está bloqueado, añadiendo nuevamente el botón de desbloquear.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	--	--
POSTCONDICIÓN	El usuario verá el tablón de seguidores con el usuario desbloqueado con el estilo de los desbloqueados. Al lado del usuario desbloqueado habrá un botón de bloquear.	

Tabla 147. Caso de uso "Desbloquear usuario".

CASO DE USO	Ver seguidos	
IDENTIFICADOR	CU-11	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario desea ver todos los usuarios a los que está siguiendo. También puede ver los 10 usuarios seguidos más activos en el tablón de usuarios en la página principal.	
PRECONDICIÓN	El usuario debe estar autenticado.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de ver todos los seguidos.	2. Obtiene todos los usuarios seguidos del usuario que realiza la acción. 3. Crea una tabla con todos los usuarios obtenidos y la muestra en una ventana.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
		1. Obtiene los 10 usuarios seguidos que más mensajes hayan escrito del usuario que realiza la acción. 2. Los muestra en el tablón de usuarios de la página principal.
POSTCONDICIÓN	El usuario verá en una ventana, todos los usuarios a los que está siguiendo. El usuario, estando en la página principal, vera los 10 usuarios que más mensajes hayan escrito de los que sigue, en el tablón de usuarios.	

Tabla 148. Caso de uso "Ver seguidos".

CASO DE USO	Ver seguidores	
IDENTIFICADOR	CU-12	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario desea ver todos los usuarios que le están siguiendo. También puede ver los 10 usuarios más activos que le siguen, en el tablón de usuarios en la página principal.	
PRECONDICIÓN	El usuario debe estar autenticado.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de ver todos los seguidores.	2. Obtiene todos los usuarios seguidores del usuario que realiza la acción. 3. Crea una tabla con todos los usuarios obtenidos y la muestra en una ventana.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
		1. Obtiene los 10 usuarios seguidores que más mensajes hayan escrito del usuario que realiza la acción. 2. Los muestra en el tablón de usuarios de la página principal.
POSTCONDICIÓN	El usuario verá en una ventana, todos los usuarios que le están siguiendo. El usuario, estando en la página principal, vera los 10 usuarios que más mensajes hayan escrito de los que le siguen, en el tablón de usuarios.	

Tabla 149. Caso de uso "Ver seguidores".

CASO DE USO	Escribir mensaje	
IDENTIFICADOR	CU-13	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere escribir un nuevo mensaje público para que sus seguidores puedan verlo.	
PRECONDICIÓN	El usuario debe estar autenticado.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario escribe el mensaje en el campo especializado para ello. 2. El usuario elige el grupo o asignatura a la que quiere añadir el mensaje. 3. El usuario envía los datos.	4. Introduce el mensaje con el autor, la fecha, el texto y el grupo. 5. Actualiza el tablón de mensajes para que aparezca el nuevo mensaje. 6. Envía un mensaje al usuario indicándole que se ha añadido un nuevo mensaje correctamente.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	6. El usuario corrige el error. 7. El usuario reenvía los datos.	4. No introduce el mensaje porque tiene más de 140 caracteres. 5. Envía un mensaje informando del error. 8. Introduce el mensaje con el autor, la fecha, el texto y el grupo. 9. Actualiza el tablón de mensajes para que aparezca el nuevo mensaje. 10. Envía un mensaje al usuario indicándole que se ha añadido un nuevo mensaje correctamente.
POSTCONDICIÓN	El usuario habrá enviado un mensaje nuevo, y podrá verlo en su tablón de mensajes del grupo correspondiente. También podrán verlo en sus respectivos tablones los usuarios seguidores del autor del mensaje.	

Tabla 150. Caso de uso "Escribir mensaje".

CASO DE USO	Borrar mensaje	
IDENTIFICADOR	CU-14	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere borrar un mensaje de los que ha escrito anteriormente.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario debe haber escrito al menos un mensaje. El usuario debe ver el mensaje que quiere borrar en el tablón de mensajes.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de borrar el mensaje, que estará en el propio mensaje. 4. El usuario confirma que quiere borrar el mensaje.	2. Obtiene el id del mensaje a borrar. 3. Envía un mensaje al usuario para que confirme que quiere borrar el mensaje. 5. Borra de la tabla de mensajes el mensaje con el id pasado. 6. Actualiza el tablón de mensajes.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	4. El usuario decide no borrar el mensaje.	5. El sistema no realiza la operación.
POSTCONDICIÓN	El usuario habrá borrado el mensaje deseado, que ya no aparecerá en el tablón de mensajes propio ni el de sus seguidores.	

Tabla 151. Caso de uso "Borrar mensaje".

CASO DE USO	Reescribir mensaje	
IDENTIFICADOR	CU-15	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere reescribir un mensaje de otro usuario para que aparezca en su tablón y lo puedan ver sus seguidores.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario debe estar siguiendo a otro usuario y que este haya escrito al menos un mensaje. El usuario debe ver el mensaje que quiere reescribir en el tablón de mensajes.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de reescribir el mensaje, que estará en el propio mensaje.	2. Obtiene el id del mensaje a reescribir. 3. Con el id pasado, obtiene la información correspondiente, autor, fecha, texto. 4. Introduce un nuevo mensaje con la información del mensaje original y añadiendo la información del autor que ha reescrito el mensaje. 5. Actualiza el tablón de mensajes.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	--	--
POSTCONDICIÓN	El usuario habrá reescrito un mensaje de otro usuario y este nuevo mensaje se podrá ver en el tablón. El nuevo mensaje será como los mensajes normales pero tendrá la información del autor y fecha del mensaje original.	

Tabla 152. Caso de uso "Reescribir mensaje".

CASO DE USO	Responder mensaje	
IDENTIFICADOR	CU-16	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere responder a un mensaje de otro usuario.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario debe estar siguiendo a otro usuario y que este haya escrito al menos un mensaje. El usuario debe ver el mensaje que quiere responder en el tablón de mensajes.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de responder, que estará en el propio mensaje. 3. El usuario escribe la respuesta. 4. El usuario envía los datos.	2. Muestra un formulario en una ventana para que el usuario escriba la respuesta. 5. Obtiene el grupo del mensaje respondido para introducir la respuesta en el mismo. 6. Introduce la respuesta en la tabla de mensajes. 7. Actualiza el tablón de mensajes.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	--	--
POSTCONDICIÓN	El usuario habrá respondido a un mensaje de otro usuario. Estos mensajes de respuestas sólo aparecerán en el tablón de los usuarios implicados.	

Tabla 153. Caso de uso " Responder mensaje".

CASO DE USO	Enviar privado	
IDENTIFICADOR	CU-17	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere enviar un mensaje privado a otro usuario.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario debe estar siguiendo a otro usuario.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón de enviar privado, en el tablón de usuarios o en un mensaje del usuario al que se quiere enviar el privado. 3. El usuario escribe el mensaje. 4. El usuario envía los datos.	2. Muestra un formulario en una ventana para que el usuario escriba el mensaje. 5. Obtiene los datos correspondientes al privado, autor, fecha, texto, etc. 6. Introduce el mensaje en la tabla de privados.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	7. El usuario corrige el error. 8. El usuario reenvía los datos.	5. Comprueba que el mensaje supera el número de caracteres permitido. 6. Envía un mensaje al usuario avisando del error. 9. Obtiene los datos correspondientes al privado, autor, fecha, texto, etc. 10. Introduce el mensaje en la tabla de privados.
POSTCONDICIÓN	El usuario habrá enviado un mensaje privado a otro usuario.	

Tabla 154. Caso de uso "Enviar privado".

CASO DE USO	Leer privado	
IDENTIFICADOR	CU-18	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere leer un privado, ya sea uno nuevo recibido, uno leído anteriormente o uno enviado por sí mismo. Accederá al tablón de privados y allí verá el que quiera.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario debe haber recibido o escrito al menos un privado.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón ver privados. 3. El usuario selecciona el tipo de privado que desea ver, recibidos, leídos o enviados. 6. El usuario pincha sobre el mensaje que desea leer.	2. Muestra una ventana con el tablón de privados. 4. Obtiene los mensajes del tipo de privado pedido. 5. Muestra la cabecera de los privados que ha obtenido (autor, fecha, asunto). 7. Obtiene la información del mensaje pedido. 8. Muestra el privado con toda su información.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
		7. Obtiene la información del privado pedido y del resto de mensajes de la conversación. (Si ha habido respuestas a este privado). 8. Muestra el privado pedido y el resto de privados de la conversación.
POSTCONDICIÓN	El usuario verá el mensaje privado que ha pedido en el tablón de privados, junto con el resto de privados de la conversación, si los hubiera.	

Tabla 155. Caso de uso "Leer privado".

CASO DE USO	Borrar privado	
IDENTIFICADOR	CU-19	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere borrar un privado de cualquiera de los tres tipos, recibido, leído o enviado.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario debe estar viendo el tablón de privados.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón borrar privado que estará en la cabecera de cualquier privado. 3. El usuario confirma que quiere borrar el mensaje.	2. Envía un mensaje para confirmar el borrado. 4. Marca como borrado el mensaje seleccionado en la BD. 5. Comprueba si lo han borrado autor y destinatario. 6. A) Si lo han borrado ambos, elimina el mensaje de la BD. B) Si sólo lo ha borrado uno, pasa a 7. 7. Elimina de la interfaz el mensaje borrado.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	3. El usuario cancela el borrado.	4. Cancela la operación.
POSTCONDICIÓN	El usuario habrá borrado el mensaje privado y no lo verá en el tablón. Si lo han borrado destinatario y autor el mensaje habrá sido eliminado de la base de datos, si sólo lo ha borrado el actor del caso de uso, ya sea el autor o el destinatario, sólo se borrará de la interfaz para que lo pueda seguir viendo la otra parte mientras no lo borre.	

Tabla 156. Caso de uso "Borrar privado".

CASO DE USO	Responder privado	
IDENTIFICADOR	CU-20	
ACTOR	Usuario	
DESCRIPCIÓN	El usuario quiere responder a un mensaje privado.	
PRECONDICIÓN	El usuario debe estar autenticado. El usuario debe haber recibido al menos un privado. El usuario debe estar viendo el tablón de privados.	
ESCENARIO NORMAL	**ACTOR**	**SISTEMA**
	1. El usuario pincha sobre el botón responder privado que estará en la cabecera de cualquier privado. 3. El usuario escribe el texto del nuevo privado. 4. El usuario envía los datos.	2. Abre en una ventana un formulario para enviar un nuevo privado. 5. Obtiene los id de los privados pertenecientes al hilo de respuestas, en caso de que los haya. 6. Introduce el nuevo privado en la base de datos con toda la información, incluido el hilo de respuestas. 7. Actualiza el tablón de privados.
ESCENARIO ALTERNATIVO	**ACTOR**	**SISTEMA**
	7. El usuario corrige el error. 8. El usuario reenvía los datos.	5. Comprueba que el mensaje supera el número de caracteres permitido. 6. Envía un mensaje al usuario avisando del error. 9. Obtiene los id de los privados pertenecientes al hilo de respuestas, en caso de que los haya. 10. Introduce el nuevo privado en la base de datos con toda la información, incluido el hilo de respuestas. 11. Actualiza el tablón de privados.
POSTCONDICIÓN	El usuario habrá respondido a un privado, y verá ese nuevo mensaje en la tabla de enviados del tablón de privados.	

Tabla 157. Caso de uso "Responder privado".